VISIT…

H. A. Eiselt
C.-L. Sandblom

Linear Programming and its Applications

Springer

Linear Programming and its Applications

H. A. Eiselt · C.-L. Sandblom

Linear Programming and its Applications

With 71 Figures
and 36 Tables

Springer

Prof. Dr. H. A. Eiselt
University of New Brunswick
Faculty of Business Administration
P.O. Box 4400
Fredericton, NB E3B 5A3
Canada
haeiselt@unb.ca

Prof. Dr. C.-L. Sandblom
Dalhousie University
Department of Industrial Engineering
P.O. Box 1000
Halifax, NS B3J 2X4
Canada
carl-louis.sandblom@dal.ca

Library of Congress Control Number: 2007931630

ISBN 978-3-540-73670-7 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilm or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springer.com

© Springer-Verlag Berlin Heidelberg 2007

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Production: LE-TEX Jelonek, Schmidt & Vöckler GbR, Leipzig
Cover-design: WMX Design GmbH, Heidelberg

SPIN 12092093 42/3180YL - 5 4 3 2 1 0 Printed on acid-free paper

“A problem well stated is a problem half solved.”

Charles Franklin Kettering

PREFACE

Based on earlier work by a variety of authors in the 1930s and 1940s, the simplex method for solving linear programming problems was developed in 1947 by the American mathematician George B. Dantzig. Helped by the computer revolution, it has been described by some as the overwhelmingly most significant mathematical development of the last century. Owing to the simplex method, linear programming (or linear optimization, as some would have it) is pervasive in modern society for the planning and control of activities that are constrained by the availability of resources such as manpower, raw materials, budgets, and time.

The purpose of this book is to describe the field of linear programming. While we aim to be reasonably complete in our treatment, we have given emphasis to the modeling aspects of the field. Accordingly, a number of applications are provided, where we guide the reader through the interactive process of mathematically modeling a particular practical situation, analyzing the consequences of the model formulated, and then revising the model in light of the results from the analysis.

Closely related to the issue of building models based on specific applications is the art of reformulating problems. Some of these models may at first appear not to be amenable to a linear representation, and we devote an entire chapter to this topic. A properly balanced treatment of linear programming will necessarily require a full discussion of both duality and postoptimality, and we dedicate one chapter to each of these two topics. As far as solution methods are concerned, we cover the simplex method as well as interior point techniques. During the last two decades, the latter have become serious challengers to the simplex method for solving large scale practical problems.

This book can be seen as the last part of a trilogy. The other two volumes have already appeared in print. "Integer Programming and Network Models" was published in 2000, and "Decision Analysis, Location Models, and Scheduling Problems" saw the light of day in 2004. All three volumes are similar in style, emphasizing models, applications and formulations/reformulations. We have also given detailed numerical illustrations for all algorithms presented, and have relied, whenever practical, on intuitive approaches. An interesting aspect is the longevity of a book like the present volume. It appears that descriptions of models keep their freshness longer than discussions of algorithms, and that references to computational aspects quickly become outdated. A statement from 1824 gives a poignant reminder of how short the life of a book may be:

> "…One thousand books are published per annum in Great Britain ... only do one hundred bring good profit ... seven hundred are forgotten in one year, one hundred in two years, ... not more than fifty survive seven years, and scarcely ten are thought of after twenty years.
>
> Of the 50,000 books published in the seventeenth century, not fifty are now in estimation; and of the 80,000 published in the eighteenth century not more than three hundred are considered worth reprinting, and not more than five hundred are sought after 1823. Since the first writings fourteen hundred years before Christ, i.e., in thirty-two centuries, only about five hundred works of writers of all nations have sustained themselves against the devouring influence of time."
>
> (Collections, Historical and Miscellaneous; and Monthly Literary Journal: edited by J. Farmer and J.B. Moore, Vol III, Concord 1824)

It is our pleasure to thank all of the people who have, in one way or another, helped to make this book a reality. Some of the typing was done by #13 (Benbin Zhang) and the figures were produced by Dong Lin. Last, but certainly not least, our sincere thanks go to Dr. Müller of Springer Publishers, whose gentle reminders kept us on track and more or less on time. We are very grateful for the assistance.

H.A. Eiselt
C.-L. Sandblom

CONTENTS

Symbols XIII

A. Linear Algebra 1
A.1 Matrix Algebra 1
A.2 Systems of Simultaneous Linear Equations 5
A.3 Convexity 23

B. Computational Complexity 31
B.1 Algorithms and Time Complexity Functions 31
B.2 Examples of Time Complexity Functions 37
B.3 Classes of Problems and Their Relations 41

1. Introduction 45
1.1. A Short History of Linear Programming 45
1.2 Assumptions and the Main Components of Linear Programming Problems 48
1.3 The Modeling Process 53
1.4 The Three Phases in Optimization 57
1.5 Solving the Model and Interpreting the Printout 60

2. Applications 67
2.1 The Diet Problem 67
2.2 Allocation Problems 71
2.3 Cutting Stock Problems 75
2.4 Employee Scheduling 80
2.5 Data Envelopment Analysis 82
2.6 Inventory Planning 85
2.7 Blending Problems 89
2.8 Transportation Problems 91
2.9 Assignment Problems 102
2.10 A Production – Inventory Model: A Case Study 107

3. The Simplex Method 129
3.1 Graphical Concepts 129
3.1.1 The Graphical Solution Technique 129
3.1.2 Four Special Cases 138
3.2 Algebraic Concepts 143
3.2.1 The Algebraic Solution Technique 143
3.2.2 Four Special Cases Revisited 158

4. Duality 167
4.1 The Fundamental Theory of Duality 167
4.2 Primal-Dual Relations 183
4.3 Interpretations of the Dual Problem 198

5. Extensions of the Simplex Method 203
5.1 The Dual Simplex Method 203
5.2 The Upper Bounding Technique 212
5.3 Column Generation 219

6. Postoptimality Analyses 225
6.1 Graphical Sensitivity Analysis 227
6.2 Changes of the Right-Hand Side Values 232
6.3 Changes of the Objective Function Coefficients 240
6.4 Sensitivity Analyses in the Presence of Degeneracy 245
6.5 Addition of a Constraint 248
6.6 Economic Analysis of an Optimal Solution 252

7. Non-Simplex Based Solution Methods 261
7.1 Alternatives to the Simplex Method 262
7.2 Interior Point Methods 273

8. Problem Reformulations 295
8.1 Reformulations of Variables 295
8.1.1 Lower Bounding Constraints 295
8.1.2 Variables Unrestricted in Sign 296
8.2 Reformulations of Constraints 298
8.3. Reformulations of the Objective Function 301
8.3.1. Minimize the Weighted Sum of Absolute Values 301
8.3.2 Bottleneck Problems 306
8.3.3 Minimax and Maximin Problems 313
8.3.4 Fractional (Hyperbolic) Programming 320

9. Multiobjective Programming 325
9.1 Vector Optimization 327
9.2 Models with Exogenous Tradeoffs Between Objectives 337
9.2.1 The Weighting Method 337
9.2.2 The Constraint Method 339
9.3 Models with Exogenous Achievement Levels 341
9.3.1 Reference Point Programming 342
9.3.2 Fuzzy Programming 346
9.3.3 Goal Programming 351
9.4 Bilevel Programming 359

References 363

Subject Index 377

SYMBOLS

This part introduces the reader to some of the support methodology used in this book. We have made every possible attempt to keep the exposition as brief and concise as possible. Readers who are interested in more in-depth coverage are referred to the pertinent literature.

Notation

$\mathbb{N} = \{1, 2, ...\}$: Set of natural numbers
$\mathbb{N}_0 = \{0, 1, 2, ...\}$: Set of natural numbers including zero
$\mathbb{R}$: Set of real numbers
$\mathbb{R}_+$: Set of nonnegative real numbers
$\mathbb{R}^n$: n-dimensional real space

$\in$: Element of
$\subseteq$: Subset
$\subset$: Proper subset
$\cup$: Union of sets
$\cap$: Intersection of sets
$\varnothing$: Empty set

$\rightarrow$: Implies
$\exists$: There exists at least one
$\forall$: For all
$|S|$: Cardinality of the set S
inf: infimum
sup: supremum

$x \in [a, b]$: $a \leq x \leq b$
$x \in [a, b[$: $a \leq x < b$
$x \in]a, b]$: $a < x \leq b$
$x \in]a, b[$: $a < x < b$

$\lceil x \rceil$: Ceiling of x, the smallest integer greater or equal to x
$\lfloor x \rfloor$: Floor of x, the largest integer smaller or equal to x
$|x|$: Absolute value of x

$a := a + b$: Valuation, a is replaced by $a + b$
$\frac{\partial f(x)}{\partial x}$: partial derivative of the function $f(x)$ with respect to x

A LINEAR ALGEBRA

The purpose of this chapter is to introduce to the reader the basic concepts of linear algebra whose knowledge is mandatory for the understanding of the material covered in the succeeding chapters. We have taken it out of the main part of the book so as not to interrupt the flow of our arguments and to let those who are familiar with the concepts discussed here, skip them and jump ahead to the Introduction in Chapter 1. With some modifications, the material in this Chapter is largely taken from Eiselt and Sandblom (2004). For a full treatment, see any text on linear algebra, e.g., Bretscher (1997) or Nicholson (2006).

A.1 Matrix Algebra

Definition A.1: A *matrix* is a two-dimensional array of elements a_{ij} arranged in m rows and n columns, so that a_{ij} is the element in row i and column j. If $m = n$, the matrix is said to be *square*; if $m = 1$, it is called a *row vector*, if $n = 1$, it is a *column vector*, and if $m = n = 1$, it is a *scalar*.

Typically, matrices are denoted by boldface capital letters, vectors are shown as boldface small letters, and scalars are shown as italicized small letters. The i-th row of a matrix $\mathbf{A} = (a_{ij})$ is $\mathbf{a}_{i\bullet}$ and the j-th column of the matrix $\mathbf{A}$ is $\mathbf{a}_{\bullet j}$.

Definition A.2: An $[n \times n]$-dimensional matrix $\mathbf{A} = (a_{ij})$ is called an *identity matrix*, if $a_{ij} = 1$ if $i = j$, and zero otherwise.

An identity matrix is denoted by $\mathbf{I}$. The i-th row of an identity matrix is called a *unit row vector* $\mathbf{e}_{i\bullet} = [0, 0, \ldots 0, 1, 0, 0, \ldots 0]$ which has the "1" in the i-th position, and zeroes otherwise. The definition of a unit column vector $\mathbf{e}_{\bullet j}$ is analogous.

Definition A.3: The *sum of* two $[m \times n]$-dimensional *matrices* $\mathbf{A}$ and $\mathbf{B}$ is an $[m \times n]$-dimensional matrix $\mathbf{C}$, such that $c_{ij} = a_{ij} + b_{ij}$ $\forall$ i=1, …, m; j=1, …, n. The difference of two matrices is defined similarly.

Definition A.4: The *product* of an $[m \times n]$-dimensional *matrix* $\mathbf{A} = (a_{ij})$ and an $[n \times p]$-dimensional matrix $\mathbf{B} = (b_{jk})$ is an $[m \times p]$-dimensional matrix $\mathbf{C} = (c_{ik})$, such that

$$c_{ik} = \sum_{j=1}^{n} a_{ij} b_{jk} \quad \forall\ i,k\ .$$

Definition A.5: The *transpose* of an $[m \times n]$-dimensional matrix $\mathbf{A} = (a_{ij})$ is an $[n \times m]$-dimensional matrix $\mathbf{A}^T = (a_{ij}^T)$, such that $a_{ij}^T = a_{ji} \ \forall\ i,j$. If $\mathbf{A} = \mathbf{A}^T$, the $\mathbf{A}$ is said to be *symmetric*. The *trace* of an $[n \times n]$-dimensional matrix $\mathbf{A}$ is defined as the sum

$$tr\ \mathbf{A} = \sum_{j=1}^{n} a_{jj}\ .$$

Definition A.6: The *inverse* of an $[n \times n]$-dimensional matrix $\mathbf{A} = (a_{ij})$, provided it exists, is a matrix $\mathbf{A}^{-1}$, such that $\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$.

Example: Let $\mathbf{A} = \begin{bmatrix} 0 & 4 & 2 \\ 0 & 3 & 2 \\ -1 & 0 & -2 \end{bmatrix}$. Then $\mathbf{A}^{-1} = \begin{bmatrix} 3 & -4 & -1 \\ 1 & -1 & 0 \\ -1½ & 2 & 0 \end{bmatrix}$.

Proposition A.7: The following results hold when multiplying, transposing, and inverting matrices **A**, **B**, and **C**:

- $\mathbf{A}\mathbf{I} = \mathbf{I}\mathbf{A} = \mathbf{A}$
- $\mathbf{A}(\mathbf{B}\mathbf{C}) = (\mathbf{A}\mathbf{B})\mathbf{C}$
- $(\mathbf{A}^T)^T = \mathbf{A}$
- $(\mathbf{A}\mathbf{B})^T = \mathbf{B}^T\mathbf{A}^T$
- $(\mathbf{A}^{-1})^{-1} = \mathbf{A}$
- $(\mathbf{A}\mathbf{B})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$
- $(\mathbf{A}^T)^{-1} = (\mathbf{A}^{-1})^T$

Definition A.8: The *determinant* of an $[n \times n]$-dimensional *matrix* $\mathbf{A}$ is defined recursively as $det\ \mathbf{A} = \sum_{j=1}^{n} (-1)^{i+j} a_{ij}\ det\ \mathbf{A}_{ij}$, where $\mathbf{A}_{ij}$ denotes the matrix that results from the given matrix $\mathbf{A}$ by deleting the i-th row and the j-th column.

This development of the determinant via the i-th row is due to Laplace. As a starting condition, the determinant of a $[1 \times 1]$-dimensional matrix $\mathbf{A}$ is $det\ \mathbf{A} =$

a_{11}. Applying Laplace's formula to a [2 × 2]-dimensional matrix $\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$, we obtain *det* $\mathbf{A} = a_{11}a_{22} - a_{12}a_{21}$. The inverse $\mathbf{A}^{-1}$ of a square matrix **A** exists if and only if *det* $\mathbf{A} \neq 0$. A more efficient technique for the evaluation of determinants is described in Procedure A.19 below and illustrated in Examples 4 and 5, following the algorithm.

Example: Let again

$$\mathbf{A} = \begin{bmatrix} 0 & 4 & 2 \\ 0 & 3 & 2 \\ -1 & 0 & -2 \end{bmatrix}.$$

Evaluating the determinant with respect to the first row, we obtain

$$\mathbf{A}_{11} = \begin{bmatrix} 3 & 2 \\ 0 & -2 \end{bmatrix}, \mathbf{A}_{12} = \begin{bmatrix} 0 & 2 \\ -1 & -2 \end{bmatrix}, \text{ and } \mathbf{A}_{13} = \begin{bmatrix} 0 & 3 \\ -1 & 0 \end{bmatrix}.$$

so that *det* $\mathbf{A} = (-1)^{1+1}(0)$ *det* $\mathbf{A}_{11} + (-1)^{1+2}(4)$ *det* $\mathbf{A}_{12} + (-1)^{1+3}(2)$ *det* $\mathbf{A}_{13} = (1)(0)(-6) + (-1)(4)(2) + (1)(2)(3) = -2$. The determinant can be interpreted as the volume of the *n*-dimensional parallelepiped spanned by the column vectors of the matrix.

Definition A.9: An *eigenvalue* or *characteristic value* λ of a square matrix **A** is a number, possibly complex, which solves the equation *det* $(\mathbf{A} - \lambda\mathbf{I}) = 0$.

One can show that if λ is a real-valued eigenvalue of the square matrix **A**, then there exists a corresponding vector $\mathbf{x} \neq \mathbf{0}$, such that $\mathbf{Ax} = \lambda\mathbf{x}$. This vector is called an *eigenvector* of **A** corresponding to the eigenvalue λ. It may happen that no real-valued eigenvalue exists. Geometrically, an eigenvalue can be interpreted as some form of "scaling" resulting from the linear transformation $\mathbf{x} \rightarrow \mathbf{Ax}$. For instance, if [1, 0] and [0, 1] are the eigenvectors for a [2×2]-dimensional matrix **A**, with eigenvalues ½ and 3, respectively, then every transformed vector **x** will have its first component halved and second component tripled.

Example: Using again the matrix **A** in the following examples of Definitions A.6 and A.8, the eigenvalues of **A** can be calculated as .47068, 2.34292, and −1.81361, and the corresponding eigenvectors are [.88867, .28441, −.35969], [.80472, .56400, −.18529], and [−.16963, −.37813, .91008], respectively.

Procedure A.10 (Newton Raphson method): The Newton-Raphson method is designed to determine roots of polynomials.

Procedure: Given the polynomial functions $f(x)$, the Newton-Raphson method for finding a real root of the polynomial, i.e., a real solution to the equation $f(x) = 0$ proceeds as follows. Beginning with an initial estimate x_1, subsequent estimates are calculated iteratively by the relation

$$x_{k+1} := x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 1, 2, \ldots$$

Example: Consider the polynomial $y = 2x^3 - 4$, whose derivative with respect to x is $f'(x) = 6x^2$. Starting with the initial guess $x_1 = 5$ for a root of this polynomial, we obtain $x_2 = 5 - \frac{2(5)^3 - 4}{6(5)^2} = 3.36$. In the subsequent iterations we obtain $x_3 = 2.2990$, $x_4 = 1.658$, $x_5 = 1.3482$, $x_6 = 1.2656$, $x_7 = 1.2599 = x_8$, and the process has converged.

Proposition A.11: The following rules apply to determinants of $[n \times n]$-dimensional matrices **A** and **B**:

$$\begin{aligned}
&det\ \mathbf{I} = 1 \\
&det\ \mathbf{AB} = det\ \mathbf{A}\ det\ \mathbf{B} \\
&det\ \mathbf{A} = det\ \mathbf{A}^T \\
&det\ \alpha\mathbf{A} = \alpha^n\ det\ \mathbf{A} \\
&det\ \mathbf{A}^{-1} = \frac{1}{det\ \mathbf{A}}.
\end{aligned}$$

Procedure A.12: Define a *simplex* S in $\mathbb{R}^d$ as a set of $(d + 1)$ points $\mathbf{x}^k$, k=1, 2, …, d+1, such that there exists no hyperplane H: $\mathbf{ax} = b$ with the property that $\mathbf{ax}^k = b$ $\forall$ k=1, 2, …, d+1. In other words, a simplex is a minimal independence structure in the sense that all but one of its extreme points are located on a hyperplane. In order to determine the volume of a d-dimensional polyhedron P, it is first necessary to subdivide P into simplices. The *volume of a simplex S* with extreme points $\mathbf{x}^k$, k = 1, 2, …, d+1 can then be expressed as

$$v(S) = \tfrac{1}{d}\ det\ \mathbf{A} = \tfrac{1}{d} det \begin{bmatrix} \mathbf{x}^1 & 1 \\ \mathbf{x}^2 & 1 \\ \vdots & \vdots \\ \mathbf{x}^{d+1} & 1 \end{bmatrix}.$$

A.2 Systems of Simultaneous Linear Equations

The purpose of this section is to provide some of the tools which are necessary for the topics in this book and thus establish a common background of knowledge. In order to do so, we will introduce those elements of linear algebra that are mandatory for the understanding of linear programming. For further details, readers are referred to texts on linear algebra, e.g., Nicholson (2006) or Bretscher (1997).

Throughout this section we assume that $x_1, x_2, \ldots, x_n$ are variables or unknowns of some given system, and $a_1, a_2, \ldots, a_n$ and b are given real numbers. The symbol R defines a relation of the type =, ≤, ≥, <, or >, and $f(\mathbf{x})$ is a function of the variables $x_1, x_2, \ldots, x_n$.

Definition A.13: A *relation* $f(\mathbf{x})$ R b is said to be *linear*, if $f(\mathbf{x}) = \sum_{j=1}^{n} a_j x_j$. We will refer to $f(\mathbf{x})$ as the left-hand side and b the right-hand side of this relation.

Example: The relation

$$\sqrt{5}x_1 + 4x_2 - 27x_3 \leq \tfrac{28}{13} \tag{1}$$

is linear, while the relations

$$\sqrt{5}x_1 + 4\sqrt{x_2} - 27x_3 \leq \tfrac{28}{13}, \tag{2}$$

$$\sqrt{5}x_1 + 4x_2^2 - 27x_3 \leq \tfrac{28}{13}, \text{ and} \tag{3}$$

$$\sqrt{5}x_1 + 4x_2x_4 - 27x_3 \leq \tfrac{28}{13} \tag{4}$$

are not, as relation (2) includes the square root of a variable, (3) has the variable x_2 raised to the second power, and (4) includes the product of two variables. Loosely speaking, a function is linear if all variables are not multiplied by other variables and appear only raised to the 0-th or first power.

For the time being, we will restrict ourselves to equations, i.e., $R = \{=\}$. Assume now that we have $m \geq 1$ linear equations and the problem is then to find a solution, i.e., an assignment of values to the variables, that satisfies all of the equations. This is referred to as a *system of simultaneous linear equations*. Consider a small example. A gardener has \$9 to spend on two types of fertilizer, *Turf King* and *Green Thumb Special*, which retail for \$1 and \$1.50 per bag, respectively. It is known that for each three bags of *Turf King*, four bags of *Green Thumb Special* are needed to offset the detrimental effects of the former. Denoting by x_1 and x_2 the number of bags of the two respective fertilizers, the system can be written as

$$1x_1 + 1.5x_2 = 9 \quad (5)$$
$$4x_1 - 3x_2 = 0, \quad (6)$$

which, incidentally, has a solution $x_1 = 3$ and $x_2 = 4$, indicating to the gardener to purchase three bags of *Turf King* and four bags of *Green Thumb Special*. As usual, the solution of a system of simultaneous linear equations provides instructions to the decision maker by way of assigning appropriate values to the variables. This is the same for linear programming problems as we will see later on in this book. Note that linearity also implies *proportionality*. The quantity of, say, *Green Thumb Special* and the total amount of money spent on that product move up together: for each unit increase in the product's quantity, the total amount of money spent on this fertilizer increases by \$1.50. This property will be destroyed if quantity discounts, economies to scale, and similar features are introduced.

It is also worthwhile to point out that often the equations specify whether the products are substitutional or complementary. In the case of substitutional goods, one product is able to more or less replace another good. Typical examples are beef and pork, potatoes and rice, buses and trains in case of transportation, houses and apartments, etc. Clearly, increased consumption of one of the goods will result in reduced consumption of any of its substitutional goods. On the other hand, in the case of complementary goods, the two products complement each other in the sense that one product needs another to be complete the product. As a result, the increased consumption of a complementary good will result in increased consumption of the other good as well. Typical examples of complementary goods are cameras and lenses, cars and fuel, guns and ammunition, tables and chairs, etc. Note that some of the "increase implications" are unidirectional, e.g., while more cars typically imply a higher fuel consumption, a higher supply of fuel will not result in the purchase of more vehicles (except in case the fuel gets cheaper as well, but even then the increase implication is tenuous).

In general, we can view each variable as adding one degree of uncertainty, while, at least in principle, each equation with its explanatory value reduces the degree of uncertainty by one. The latter is, however, not always the case. Consider again the above system of simultaneous linear equations (5) and (6), and add the equation $5x_1 - 1.5x_2 = 9$ to the system. It can be shown that there is actually no new information contained in this equation as it is generated as the sum of (5) and (6). Below we will develop conditions that indicate when an equation adds new information to the system and when this is not the case.

In order to formalize, consider the system of simultaneous linear equations $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is an $[m \times n]$-dimensional matrix, $\mathbf{x}$ is an n-dimensional column vector, and $\mathbf{b}$ is an m-dimensional column vector. In terms of the above example, $\mathbf{A} = \begin{bmatrix} 1 & 1.5 \\ 4 & -3 \end{bmatrix}$, $\mathbf{x} = [x_1, x_2]^T$, and $\mathbf{b} = [9, 0]^T$. Let now $\mathbf{a}_{i\bullet}$ denote the i-th row of

the matrix **A** in accordance with Definition A.1, so that in our example, $\mathbf{a}_{1\bullet} = [1, 1.5]$ and $\mathbf{a}_{2\bullet} = [4, -3]$.

Definition A.14: A set of vectors $\mathbf{a}_{1\bullet}, \mathbf{a}_{2\bullet}, \ldots, \mathbf{a}_{m\bullet}$ is said to be *linearly dependent*, if there exists a vector $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_m] \neq \mathbf{0}$ of real numbers, such that $\sum_{i=1}^{m} \lambda_i \mathbf{a}_{i\bullet} = 0$. If the vectors $\mathbf{a}_{1\bullet}, \mathbf{a}_{2\bullet}, \ldots, \mathbf{a}_{m\bullet}$ are not linearly dependent, they are said to be *linearly independent*.

It is not difficult to demonstrate that if the vectors $\mathbf{a}_{1\bullet}, \mathbf{a}_{2\bullet}, \ldots, \mathbf{a}_{m\bullet}$ are linearly dependent, then there exists some number k, $1 \leq k \leq m$ and some real numbers λ_i, such that $\mathbf{a}_{k\bullet} = \sum_{\substack{i=1 \\ k \neq 1}}^{m} \lambda_i \mathbf{a}_{i\bullet}$. In other words, at least one of the vectors in the system can be generated as a weighted sum of the others. It also follows that if the vectors are linearly independent, then $\sum_{i=1}^{m} \lambda_i \mathbf{a}_{i\bullet} = 0$ implies that $\lambda_i = 0 \ \forall \ i = 1, 2, \ldots, m$. A similar definition and results can be developed for column vectors. As an example, consider the matrix

$$\mathbf{A} = \begin{bmatrix} 2 & -3 \\ 5 & 1 \\ -4 & -11 \\ 9 & -5 \end{bmatrix}.$$

Here, the first row can be expressed as $\mathbf{a}_{1\bullet} = [2, -3] = \lambda_2 \mathbf{a}_{2\bullet} + \lambda_3 \mathbf{a}_{3\bullet} + \lambda_4 \mathbf{a}_{4\bullet} = \lambda_2 [5, 1] + \lambda_3 [-4, -11] + \lambda_4 [9, -5]$ with $\lambda_2 = ⅔$, $\lambda_3 = ⅓$, and $\lambda_4 = 0$. In other words, a weighted combination of the second and third rows can generate the first row, so that row 1 does not include any information beyond what is provided by rows 2, 3, and 4. Using the same argument, the fourth row can be written as $\mathbf{a}_{4\bullet} = [9, -5] = \lambda_2 \mathbf{a}_{2\bullet} + \lambda_3 \mathbf{a}_{3\bullet} = \lambda_2 [5, 1] + \lambda_3 [-4, -11]$ with $\lambda_2 = \frac{7}{3}$ and $\lambda_3 = \frac{2}{3}$, so that the second and third rows can be used to generate the fourth row, which, again, does not provide any information beyond what is already included in rows 2 and 3. While we have just shown that rows 2 and 3 contain as much information as the four rows combined, it is also true that rows 1 and 2 can be used to generate rows 3 and 4, respectively, so they also contain all the information included in the system. The message here is that different combinations of rows may include all the information contained in the system. We will now formalize the discussion.

For that purpose, we can write

Definition A.15: The *rank* of a matrix **A**, written *rk* **A**, is the maximal number of linearly independent rows and columns of **A**.

For instance, in the above example *rk* **A** = 2. It should be noted that in an $[m \times n]$-dimensional matrix the rank is not necessarily equal to min $\{m, n\}$, but we certainly have *rk* **A** ≤ min $\{m, n\}$. If, however, *rk* **A** = min $\{m, n\}$, we say that the matrix **A** has *full rank*.

Let now **A** be an $[m \times n]$-dimensional matrix, so that m equations and n variables are included in the system **Ax** = **b**, and let [**A**, **b**] denote the matrix that includes all columns of **A** as well as the vector **b**. Clearly, *rk* [**A**, **b**] ≥ *rk* **A**. We can now state the following

Theorem A.16: A system of simultaneous linear equations **Ax** = **b** has

(*i*) no solution if *rk* **A** < *rk* [**A**, **b**],
(*ii*) exactly one solution, if *rk* **A** = *rk* [**A**, **b**] = n, and
(*iii*) an infinite number of solutions, if *rk* **A** = *rk* [**A**, **b**] < n.

A proof of Theorem A.16 can be found in many books on linear algebra; see, e.g., Nicholson (2006) or Bretscher (1997). The following examples may explain the three cases outlined in the theorem.

Example 1: Let $\mathbf{A} = \begin{bmatrix} 8 & 4 & 12 \\ 3 & 7 & 6 \\ 11 & 22 & 21 \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} 8 \\ 5 \\ 20 \end{bmatrix}$.

Then *rk* **A** = 2, since $\mathbf{a}_{3\bullet} = ¼\mathbf{a}_{1\bullet} + 3\mathbf{a}_{2\bullet}$, but there is no λ_2, such that $\mathbf{a}_{1\bullet} = \lambda_2\mathbf{a}_{2\bullet}$. On the other hand,

$$rk\,[\mathrm{A, b}] = rk \begin{bmatrix} 8 & 4 & 12 & 8 \\ 3 & 7 & 6 & 5 \\ 11 & 22 & 21 & 20 \end{bmatrix} = 3,$$

since none of the three rows can be generated by the two other rows of the matrix. Hence, the above system has no solution.

Example 2: Let $\mathbf{A} = \begin{bmatrix} 1 & 3 \\ 2 & 7 \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} 2 \\ 5 \end{bmatrix}$. Here, *rk* **A** = 2 and

$$rk\,[\mathbf{A}, \mathbf{b}] = rk \begin{bmatrix} 1 & 3 & 2 \\ 2 & 7 & 5 \end{bmatrix} = 2,$$

so that the system will have exactly one solution, which, incidentally, is $x_1 = -1$ and $x_2 = 1$.

Example 3: Let $\mathbf{A} = \begin{bmatrix} 1 & 3 & 4 \\ 2 & 7 & 1 \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} 2 \\ 5 \end{bmatrix}$. Now $rk\,\mathbf{A} = 2$ and

$$rk\,[\mathbf{A}, \mathbf{b}] = rk \begin{bmatrix} 1 & 3 & 4 & 2 \\ 2 & 7 & 1 & 5 \end{bmatrix} = 2 < 3 = n.$$

Thus, there are infinitely many solutions to this system, e.g., $\mathbf{x} = [-1, 1, 0]^T$, $\mathbf{x} = \left[\frac{18}{7}, 0, -\frac{1}{7}\right]^T$, $\mathbf{x} = \left[\frac{1}{3}, \frac{47}{75}, -\frac{4}{75}\right]^T$, etc.

Definition A.17: Let $\mathbf{A}$ be an $[m \times n]$-dimensional matrix with $n \geq m$. Any collection of m linearly independent columns of $\mathbf{A}$ is said to be a *basis* of $\mathbf{A}$.

It follows that if a matrix $\mathbf{A}$ has full rank, i.e., if $rk\,\mathbf{A} = m$, then $\mathbf{A}$ has at least one basis. Consider now an $[m \times n]$-dimensional matrix $\mathbf{A}$ with $n \geq m$ and full rank and one of its bases. The columns forming this basis constitute an $[m \times m]$-dimensional submatrix $\mathbf{B}$, which we will call a *basis matrix*. We can then partition the original matrix $\mathbf{A} = [\mathbf{B}, \mathbf{N}]$, maybe after reordering the columns, where the $[m \times (n-m)]$-dimensional matrix $\mathbf{N}$ consists of all columns of $\mathbf{A}$ that are not in the basis. The vector of variables $\mathbf{x}$ can be partitioned accordingly into $\mathbf{x}^T = [\mathbf{x}_\mathbf{B}^T, \mathbf{x}_\mathbf{N}^T]$.

The components of the m-dimensional vector $\mathbf{x_B}$ are called *basic variables*, whereas the components of the $(n-m)$-dimensional vector $\mathbf{x_N}$ are called *nonbasic variables*. Then the system $\mathbf{Ax} = \mathbf{b}$ can be written $\mathbf{Bx_B} + \mathbf{Nx_N} = \mathbf{b}$. Setting $\mathbf{x_N} := \mathbf{0}$, we see that this system has a solution $\mathbf{x_B} = \mathbf{B}^{-1}\mathbf{b}$, $\mathbf{x_N} = \mathbf{0}$. The concept of the basis of a matrix is of fundamental importance in linear programming and it will be discussed further below.

Looking at the graphical representation in two dimensions, it is well known that each equation can be represented by a straight line. The combinations of all coordinates (x_1, x_2) on the line satisfy this equation. Given two equations, the two lines are either parallel (in which case there is no pair of coordinates that satisfies both of them, i.e., there is no solution), they intersect at a single point (in which case there is a unique solution), or they coincide (in which case any point on the two identical lines is a solution, i.e., there is an infinite number of solutions). These are exactly the three cases (*i*), (*ii*), and (*iii*) outlined in Theorem A.16 above.

In the following we will derive a method that is not only efficient in finding solutions for systems of simultaneous linear equations, but it is also at the core of the standard method for a solution technique for linear programming problems. In fact, there are a number of solution methods for the solution of systems of simultaneous linear equations, e.g., substitution, Gaussian elimination, or Cramer's rule, just to name a few. Let us first consider

Procedure A.18 (Cramer's rule): Consider the system of simultaneous linear equations $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is an $[n \times n]$-dimensional nonsingular matrix, so that $det\ \mathbf{A} \neq 0$. Then the unique solution to the system is obtained by the formula

$$x_j = \frac{det\ \mathbf{A}_j}{det\ \mathbf{A}}, \text{ where}$$

$$\mathbf{A}_j = \begin{bmatrix} a_{11} & a_{12} & \cdots & b_1 & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & b_2 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & b_n & \cdots & a_{nn} \end{bmatrix},$$

i.e., the matrix $\mathbf{A}_j$ is the matrix $\mathbf{A}$ with the j-th column $\mathbf{a}_{\bullet j}$ being replaced by the right-hand side vector $\mathbf{b}$.

Example: Consider the following system of simultaneous linear equations:

$$\begin{aligned} x_1 + 4x_2 &= 5 \\ -2x_1 + 6x_2 &= -3. \end{aligned}$$

With $\mathbf{A} = \begin{bmatrix} 1 & 4 \\ -2 & 6 \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} 5 \\ -3 \end{bmatrix}$, we obtain $det\ \mathbf{A} = 14$, so that

$$x_1 = \tfrac{1}{14}\, det \begin{bmatrix} 5 & 4 \\ -3 & 6 \end{bmatrix} = \tfrac{1}{14}(42) = 3,$$

$$x_2 = \tfrac{1}{14}\, det \begin{bmatrix} 1 & 5 \\ -2 & -3 \end{bmatrix} = \tfrac{1}{14}(7) = ½,$$

so that we obtain the solution $\mathbf{x} = \begin{bmatrix} 3 \\ ½ \end{bmatrix}$.

Procedure A.19 (Gauss-Jordan pivoting method): The Gauss-Jordan pivoting method can be considered an "automated" implicit substitution method. To

initialize, consider the system $\sum_{j=1}^{n} a_{ij} x_j = b_i$, i=1, 2, …, m. Choose now any equation, say the r-th, and any variable, say the s-th, so that the coefficient of the variable x_s in the r-th equation is a_{rs}, which we assume to be nonzero. The r-th equation can then be written as $a_{rs} x_s + \sum_{\substack{j=1 \\ j \neq s}}^{n} a_{rj} x_j = b_r$ or as

$$x_s + \sum_{\substack{j=1 \\ j \neq s}}^{n} \frac{a_{rj}}{a_{rs}} x_j = \frac{b_r}{a_{rs}}. \tag{7}$$

Rewriting (7) results in $x_s = \frac{b_r}{a_{rs}} - \sum_{\substack{j=1 \\ j \neq s}}^{n} \frac{a_{rj}}{a_{rs}} x_j$. Substituting x_s in all equations $i \neq r$ results in

$$a_{is} x_s + \sum_{\substack{j=1 \\ j \neq s}}^{n} a_{ij} x_j = b_i \text{ , or } \frac{a_{is} b_r}{a_{rs}} - \sum_{\substack{j=1 \\ j \neq s}}^{n} \frac{a_{is} a_{rj}}{a_{rs}} x_j + \sum_{\substack{j=1 \\ j \neq s}}^{n} a_{ij} x_j = b_i \text{ ,}$$

or simply

$$\sum_{\substack{j=1 \\ j \neq s}}^{n} \left(a_{ij} - \frac{a_{is} a_{rj}}{a_{rs}} \right) x_j = b_i - \frac{a_{is} b_r}{a_{rs}} \quad \forall\ i \neq r. \tag{8}$$

The original system has now been transformed into a new system that consists of the single equation (7) and the (m − 1) equations (8). Note that now the variable x_s appears with a column that has a coefficient of 1 in the r-th row and with coefficients of 0 in all other rows. This means that at present, $x_s = b_r$, where b_r refers to the numerical value that is presently on the right-hand side in row r. Repeated applications of the above procedure result in each row having only a single "1" coefficient on the left-hand side of the equation while all other coefficients are zero. This allows us to directly read the solution from the system.

This procedure can be performed in a systematic manner as described below. We initialize the method by defining the two set of indices I:= {1, 2, …, m} and J := {1, 2, …, n}.

The Gauss-Jordan Pivoting Method

Step 1: Does there exist a row $i \in I$, such that $\mathbf{a}_{i\bullet} = 0$?
If yes: Go to Step 2.
If no: Go to Step 3.

Step 2: Is $b_i = 0$?
If yes: Set $I := I \setminus \{i\}$, delete the i-th row and go to Step 1.
If no: Stop, the system has no solution.

Step 3: Is $I = \varnothing$?
If yes: Go to Step 4.
If no: Go to Step 5.

Step 4: Is $J = \varnothing$?
If yes: Stop, the system has exactly one solution $x_j = b_i$ for $\mathbf{a}_{\bullet j} = \mathbf{e}_i$.
If no: Stop, the system has an infinite number of solutions, one of which is $x_j = b_i$ for $\mathbf{a}_{\bullet j} = \mathbf{e}_i$ and $x_j = 0 \ \forall \ j \in J$.

Step 5: Select a *pivot element* (also simply called a pivot) $a_{rs} \neq 0$, such that $r \in I$ and $s \in J$. Then calculate

$$a_{ij} := \begin{cases} \left.\begin{array}{l} 1, \text{ if } i = r \text{ and } j = s \\ 0, \text{if } i \neq r \text{ and } j = s \end{array}\right\} \text{ pivot column} \\ \left.\dfrac{a_{ij}}{a_{rs}}, \text{ if } i = r \text{ and } j \neq s\right\} \text{ pivot row} \\ \left.a_{ij} - \dfrac{a_{rj}a_{is}}{a_{rs}}, \text{ if } i \neq r \text{ and } j \neq s\right\} \text{ all other elements} \end{cases}$$

$$b_i := \begin{cases} \dfrac{b_r}{a_{rs}}, \text{ if } i = r \\ b_i - \dfrac{b_r a_{is}}{a_{rs}}, \text{ if } i \neq r \end{cases}$$

and set $I := I \setminus \{r\}$ and $J := J \setminus \{s\}$. Go to Step 1.

For manual calculations, it is useful to observe that if $a_{rj} = 0$ for some $j \neq s$, then $\mathbf{a}_{\bullet j}$ remains unchanged in the present iteration (i.e., the j-th column does not change) and, equivalently, if $a_{is} = 0$ for some $i \neq r$, then $\mathbf{a}_{i\bullet}$ remains unchanged in the present iteration (i.e., the i-th row remains unchanged).

The following three examples may explain the Gauss-Jordan pivoting method. In the tables, the circled elements denote the pivot elements.

Example 1: Consider the following system of simultaneous linear equations

$$\begin{aligned} x_1 + 2x_2 + 3x_3 &= 1 \\ 4x_1 + 5x_2 + 6x_3 &= 2 \\ 7x_1 + 8x_2 + 9x_3 &= 4. \end{aligned}$$

The system and the Gauss-Jordan iterations can then be shown in tabular form as follows:

$T^{1:}$

x_1	x_2	x_3	1
(1)	2	3	1
4	5	6	2
7	8	9	4

Initially, $I = \{1, 2, 3\}$ and $J = \{1, 2, 3\}$ and the table represents the original system of simultaneous linear equations that are read by multiplying each element in a row by its header, where the vertical line to the left of the rightmost column represents the "=". In the table above, each of the nine elements on the left-hand side can be selected as a pivot. The selection of a_{11} leads to the second table below and the modified sets $I = \{2, 3\}$ and $J = \{2, 3\}$, so that in the next iteration, only a_{22}, a_{23}, a_{32}, and a_{33} can be selected as pivot.

T^{2}:

x_1	x_2	x_3	1
1	2	3	1
0	(3)	–6	–2
0	–6	–12	–3

The selection of the element a_{22} as a pivot leads to the third table and the modified sets $I = \{3\}$ and $J = \{3\}$. Performing another iteration leads to the third table below.

T^{3}:

x_1	x_2	x_3	1
1	0	–1	–⅓
0	1	2	⅔
0	0	0	1

At this point, row 3 contains only zeroes on the left-hand side and "1" on the right-hand side, reading 0 = 1, an obvious contradiction. The conclusion is that the system has no feasible solution.

Example 2: Consider the following system of simultaneous linear equations:

$$\begin{aligned} x_1 + 4x_2 &= 5 \\ 3x_1 - 2x_2 &= 8 \\ -2x_1 + 6x_2 &= -3. \end{aligned}$$

The Gauss-Jordan pivoting steps are shown below in a table with three parts separated by horizontal lines; as usual, the pivots are shown as circled elements.

x_1	x_2	1
(1)	4	5
3	–2	8
–2	6	–3
1	4	5
0	(–14)	–7
0	14	7
1	0	3
0	1	½
0	0	0

The pivots in the first two parts of the table are chosen according to the standard criteria; in the third part, $\mathbf{a}_{3\bullet} = \mathbf{0}$ and $b_3 = 0$, so that the third row shows the obvious identity $0 = 0$. The algorithm terminates with the unique solution $x_1 = 3$ and $x_2 = ½$.

Example 3: Consider the following system of simultaneous linear equations

$$\begin{aligned} 3x_1 - 7x_2 + x_3 &= 3 \\ -2x_1 + 5x_2 &= 6 \end{aligned}$$

Again, the table below shows the steps in tabular form with the pivots circled.

x_1	x_2	x_3	1
(3)	–7	1	3
–2	5	0	6
1	$(-\frac{7}{3})$	$\frac{1}{3}$	1
0	$\frac{1}{3}$	$\frac{2}{3}$	8
1	0	5	57
0	1	2	24

As can be seen in the above table, after two steps, $I = \varnothing$ but $J \neq \varnothing$, so that the system has an infinite number of solutions. One of these solutions is $x_1 = 57$, $x_2 = 24$ and $x_3 = 0$. Other solutions can easily be generated by setting any one of the variables to an arbitrary value and solving the system for the remaining ones. For

instance, setting $x_3 := 10$, the bottom two rows of the table read $x_1 + 5(10) = 57$ and $x_2 + 2(10) = 24$, so that $x_1 = 7$ and $x_2 = 4$.

It is also worth mentioning that in case $m \geq n$, the system may have either none, one, or an infinite number of solutions, while for $m < n$, the system has either no solution or an infinite number of solutions, but never a unique solution.

Another possibility is to work with multiple right-hand sides. Such scenarios may derive from the existence of potentially different capacities, demands, or similar data, or the uncertainty surrounding these data. The use of multiple right-hand side values can easily accommodated by pivoting on them as one would do with any column in the tableau.

We should also point out that the Gauss-Jordan pivoting method described above has other uses as well. One such use is the evaluation of determinants. For that purpose, assume that the matrix **A** is of size $[n \times n]$. The procedure will then pivot on the matrix **A** (without the right-hand sides) as usual. When the algorithm terminates after exactly n pivots have been selected and the matrix has been transformed exactly n times, then *det* **A** equals the absolute value of the product of the pivots used in the procedure. If the algorithm terminates earlier (e.g., by the inability to find a nonzero pivot element), then *det* **A** = 0, which indicates that at least one row or column is linearly dependent. The following two examples may illustrate the procedure.

Example 4: Consider the matrix

$$\mathbf{A} = \begin{bmatrix} 2 & 0 & 1 \\ 3 & 1 & 2 \\ 0 & 2 & 1 \end{bmatrix}.$$

The procedure then determines the sequence of pivots shown by the circled elements in the table below.

2	0	(1)
3	1	2
0	2	1
2	0	1
–1	1	0
(–2)	2	0
0	2	1
0	0	0
1	–1	0

The third and bottom part of the table indicates that the second row, which is the only choice as pivot row at that time, consist of only zeroes, so that no pivot

selection is possible. Hence the algorithm terminates prematurely with the message that *det* **A** = 0. Furthermore, since two steps have been performed, we know that *rk* **A** = 2 and one of the rows/columns is linearly dependent on the others.

Example 5: Consider the matrix

$$\mathbf{A} = \begin{bmatrix} 2 & 0 & 1 \\ 3 & 1 & 2 \\ 0 & 1 & -7 \end{bmatrix}$$

Again, the Gauss-Jordan pivoting steps are shown in the table below.

2	0	1
(3)	1	2
0	1	−7
0	$-\frac{2}{3}$	$-\frac{1}{3}$
1	$\frac{1}{3}$	$\frac{2}{3}$
0	1	(−7)
0	$(-\frac{5}{7})$	0
1	$\frac{3}{7}$	0
0	$-\frac{1}{7}$	1
0	1	0
1	0	0
0	0	1

There were $n = 3$ pivot steps in this example, indicating that the matrix has full rank and the determinant is $|det\ \mathbf{A}| = (3)(-7)(-\frac{5}{7}) = 15$. Since two rows of the bottom part of the table have to be exchanged in order to obtain an identity matrix, the sign of the determinant changes, so that *det* **A** = 15.

Finally, we will show how to employ Gauss-Jordan pivoting to determine the inverse of a given matrix **A** (or to show that it does not exist). Suppose again that **A** is an $[n \times n]$-dimensional matrix with full rank, i.e., *det* $\mathbf{A} \neq 0$. (If this is not originally the case, it will be detected during the execution of the algorithm by a row of zeroes). Instead of just one right-hand side column, the right-hand side in this case consists of an $[n \times n]$-dimensional identity matrix **I**. The procedure will sequentially select n pivot elements in the matrix **A**, one at a time, so that the transformation rules in Step 5 of the algorithm are equally applied to the left-hand side and the right-hand side. After n pivoting steps, the matrix **A** on the left-hand side is transformed to an identity matrix (after possibly rearranging rows), while the former identity matrix on the right-hand side is transformed to the inverse $\mathbf{A}^{-1}$.

The procedure may be visualized as follows:

A	**I**
.	.
.	.
.	.
.	.
.	.
I	$\mathbf{A}^{-1}$

Again, if not all pivots were selected on the main diagonal of A, it may be necessary to exchange rows in the bottom table. clearly, if any two rows on the left hand-side are exchanged, the same change has to be performed on the right-hand side. A byproduct of this inversion is again the determinant of the matrix **A** (and $det\ \mathbf{A}^{-1} = [det\ \mathbf{A}]^{-1}$). The procedure may be explained by

Example 6: Consider the matrix

$$\mathbf{A} = \begin{bmatrix} 3 & 4 & 2 \\ 5 & 0 & -1 \\ -3 & 2 & 0 \end{bmatrix}.$$

The Gauss-Jordan pivoting steps are shown in the table below.

3	4	2	1	0	0
5	0	(−1)	0	1	0
−3	2	0	0	0	1
13	4	0	1	2	0
−5	0	1	0	−1	0
−3	(2)	0	0	0	1
(19)	0	0	1	2	−2
−5	0	1	0	−1	0
$-\frac{3}{2}$	1	0	0	0	$\frac{1}{2}$
1	0	0	$\frac{1}{19}$	$\frac{2}{19}$	$-\frac{2}{19}$
0	0	1	$\frac{5}{19}$	$-\frac{9}{19}$	$-\frac{10}{19}$
0	1	0	$\frac{3}{38}$	$\frac{3}{19}$	$\frac{13}{38}$

In order to obtain an identity matrix on the left side, the second and third row in the bottom part of the table have to be exchanged. The result is then the inverse

$$\mathbf{A}^{-1} = \begin{bmatrix} \frac{1}{19} & \frac{2}{19} & -\frac{2}{19} \\ \frac{3}{38} & \frac{3}{19} & \frac{13}{38} \\ \frac{5}{19} & -\frac{9}{19} & -\frac{10}{19} \end{bmatrix}.$$

A byproduct is the determinant of the matrix **A**, which turns out to be *det* **A** = 38, since the product of the pivots is (–1)(2)(19) = –38 and the last two rows have been exchanged, thus changing the sign.

Below, we will introduce some new terminology which is necessary for our discussion of linear relations in *n* dimensions.

Definition A.20: The set of points $\{\mathbf{x}: \mathbf{a}_{i\bullet}\mathbf{x}\ R\ b_i\}$ defines

(i) a *hyperplane*, if $R = \{=\}$,
(ii) a *closed halfspace*, if $R \in \{\leq, \geq\}$, or
(iii) an *open halfspace*, if $R \in \{<, >\}$.

In the case of a single dimension, i.e., if a single variable is given, a hyperplace is a point (e.g., $x = 4$), while a halfspace is a halfline (e.g., $x \leq 4$ or $x \geq 4$). In two dimensions a hyperplane defines a straight line and a halfspace half of a plane. Finally, in three dimensions a hyperplane defines a plane and a halfspace half of the three-dimensional space bordered by a hyperplane.

Generally speaking, a hyperplane in *n*-dimensional real space $\mathbb{R}^n$ is $(n-1)$-dimensional, while a halfspace in $\mathbb{R}^n$ is *n*-dimensional. Note that every halfspace $\mathbf{a}_{i\bullet}\mathbf{x}\ R\ b_i$ with $R \in \{\leq, \geq, <, >\}$ is bordered by the hyperplane $\mathbf{a}_{i\bullet}\mathbf{x} = b_i$, where a closed halfspace includes all points of its bordering hyperplane, while an open halfspace includes none of them. Formally we can write

Definition A.21: A *set S* is called *closed*, if all points on the boundary of *S* belong to the set as well. On the other hand, a *set S* is called *open*, if no point on the boundary of *S* belongs to *S*.

As an example, the set $\{x \in \mathbb{R}: x \in [1, 5]\}$ is closed, while the set $\{x \in \mathbb{R}: x \in]1, 5[\}$ is open. We should note here that linear programming exclusively deals with closed sets. Hence, if we simply refer to halfspaces in linear programming, this means closed halfspaces.

Consider now the case of more than one single inequality. Equivalent to the above discussion regarding equations, a point that satisfies all inequalities will graphically be located in the intersection of the halfspaces that are defined by the given inequalities.

As an illustration, consider the system of linear inequalities

$$3x_1 + 2x_2 \leq 6 \qquad (I)$$
$$2½x_1 - 1x_2 \geq 1. \qquad (II)$$

Figure A.1 shows the straight lines *I* and *II* representing the equations $3x_1 + 2x_2 = 6$ and $2½x_1 - 1x_2 = 1$, while the flags at the ends of the lines indicate the halfspace defined by the relations. These halfspaces subdivide the two-dimensional plane into the four sets *A*, *B*, *C* and *D*. All points in the set *A* violate constraint *I* and satisfy constraint *II*, all points in *B* violate both constraints, all points in the set *C* satisfy constraint *I* and violate constraint *II*, and all points in the set *D* (including the boundaries) satisfy both constraints. This is why *D* would be called the *feasible set*, or, equivalently, the *set of feasible solutions*.

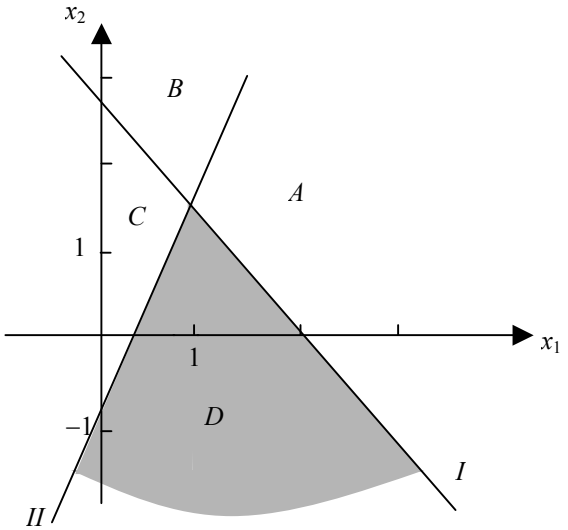


Figure A.1

Definition A.22: The *nonnegative orthant* $\mathbb{R}^n_+$ is the set of all nonnegative points in $\mathbb{R}^n$, i.e., $\bigcap_{j=1}^{n} \{x_j : x_j \geq 0\}$ or simply $\{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} \geq \mathbf{0}\}$.

In $\mathbb{R}^2$, the nonnegative orthant is the first quadrant including the positive axes and the origin. If we were to add the relations *III*: $x_1 \geq 0$ and *IV*: $x_2 \geq 0$ to the above system of simultaneous linear inequalities, then the set of solutions is the intersection of the set *D* in Figure A.1 and the first quadrant, i.e. the triangle with the vertices (2/5, 0), (2, 0), and (1, 1½).

Definition A.23: A set S is said to be *bounded*, if there exists a finite number $c \in \mathbb{R}$, so that $\|\mathbf{x}\| < c$ for every point $\mathbf{x} \in S$. A set that is not bounded is said to *unbounded*. A set S is called *compact*, if it is closed and bounded.

As an example, the set D in Figure A.1. is unbounded, while the set of points that satisfies the constraints *I*, *II*, *III*, and *IV* is bounded and closed and hence compact.

Definition A.24: The intersection of a finite number of hyperplanes and/or closed halfspaces in $\mathbb{R}^n$ is called a *polytope*. A bounded polytope is called a *polyhedron*.

In general, we will refer to the set that is defined by a system of simultaneous linear inequalities of type ≤ and/or ≥ as a polytope since it is not obvious whether or not the set is bounded. Also, a polytope and a polyhedron may degenerate to the empty set or a single point in $\mathbb{R}^n$. Again, the set D in Figure A.1 is a polytope since it is generated by the halfspaces of constraints *I* and *II*, but it is not a polyhedron, since it is unbounded from below. On the other hand, the set of point in the triangle with vertices (2/5, 0), (2, 0), and (1, 1½) is a polytope since it is generated by the halfspaces of constraints *I*, *II*, *III*, and *IV*, and since it is bounded, it is also a polyhedron. We wish to point out that the terms polytope and polyhedron are sometimes defined differently by other authors.

Let now H_i denote the set of points that satisfy the i-th linear relation $\sum_{j=1}^{n} a_{ij} x_j \;\; R_i \;\; b_i$, $i = 1, 2, \ldots, m$; then $S = \bigcap_{i=1}^{m} H_i$ is the set of points that satisfy all m relations simultaneously.

Definition A.25: The k-th relation is said to be *redundant*, if $\bigcap_{\substack{i=1 \\ i \neq k}}^{m} H_i = S.$ If the k-th relation is not redundant, i.e., if $\bigcap_{\substack{i=1 \\ i \neq k}}^{m} H_i \underset{\neq}{\supset} \bigcap_{i=1}^{m} H_i = S$, it is called *essential*.

Moreover, we can state

Definition A.26: If for some point $\tilde{\mathbf{x}} \in S$ the i-th linear relation is satisfied as an equation, i.e., if $\mathbf{a}_{i\bullet} \tilde{\mathbf{x}} = b_i$, then the i-th relation is said to be *tight* or *binding* at $\tilde{\mathbf{x}}$.

Figures A.2a and A.2b may explain the concept.

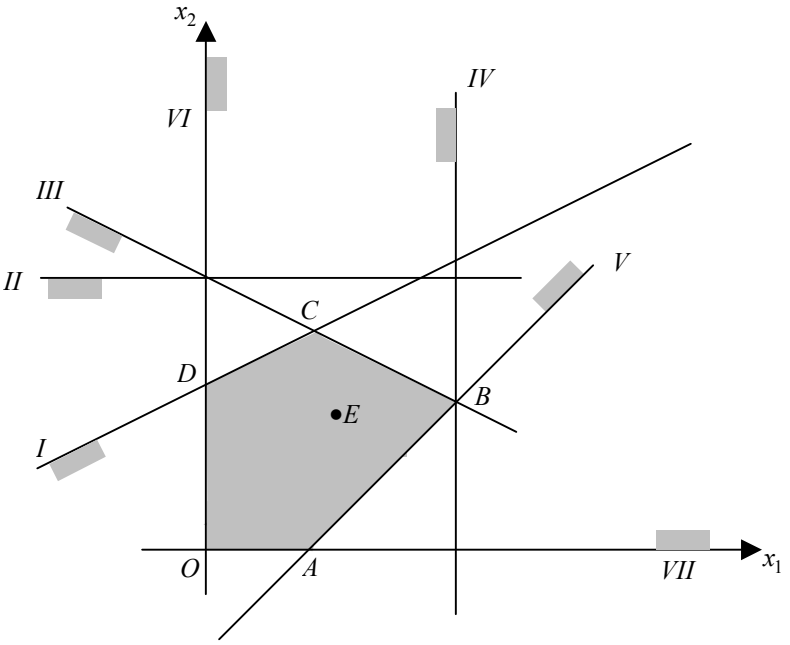


Figure A.2a

In Figure A.2a, the polytope generated by the halfspaces *I*, *II*, …, *VII* is the shaded area with corner points *0*, *A*, *B*, *C*, and *D*. Clearly, inequality *II* is redundant since its inclusion (or removal) does not change the shape of the polytope. The same argument can be applied to inequality *IV*, which therefore is also redundant. Note, however, that inequality *IV* is binding at point *B*, whereas inequality *II* is not binding at any point of the polytope, which makes constraint *II strongly redundant*, whereas constraint *IV* is *weakly redundant*. At point *C* in Figure A.2a, relations *I* and *III* are binding, at point *0* it is relations *VI* and *VII* that are binding, whereas at the interior point *E* none of the relations is binding.

In Figure A.2b, relations *I* and *II* are inequalities while relation *III* is an equation, thus the polytope defined by the relations is the set of points on the straight line between the points *A* and *B*. This implies that relations *I* and *V* are redundant and could be deleted without changing the polytope.

Definition A.27: A point $\mathbf{y} \in \mathbb{R}^n$ is said to be a *linear combination* of a given set of points $\mathbf{x}^1$, $\mathbf{x}^2$, …, $\mathbf{x}^r$ if there exist real numbers λ_1, λ_2, …, λ_r, such that $\mathbf{y} = \sum_{k=1}^{r} \lambda_k \mathbf{x}^k$. The linear combination $\mathbf{y} = \sum_{k=1}^{r} \lambda_k \mathbf{x}^k$ is said to be a *nonnegative linear combination*, if $\lambda_k \geq 0 \;\forall\; k = 1, \ldots, r$; it is called an *affine linear*

combination, if $\sum_{k=1}^{r} \lambda_k = 1,$ and it is called a *linear convex combination* (*lcc*) if $\lambda_k \geq 0 \; \forall \; k = 1, \ldots, r$, and $\sum_{k=1}^{r} \lambda_k = 1.$

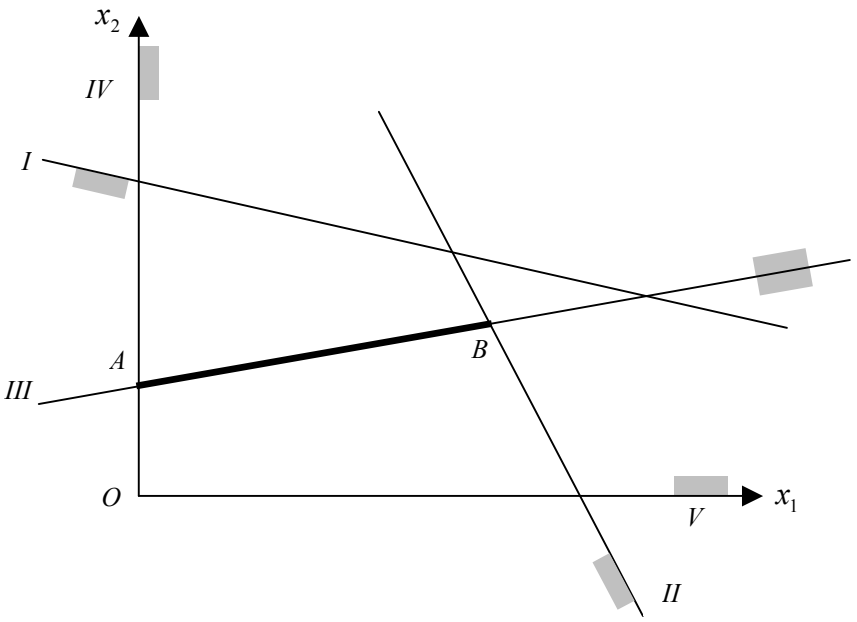


Figure A.2b

As an illustration, consider the five points $\mathbf{x}^1 = (0, 0)$, $\mathbf{x}^2 = (3, 0)$, $\mathbf{x}^3 = (0, 2)$, $\mathbf{x}^4 = (3, 2)$, and $\mathbf{y} = (1½, ½)$. Inspection reveals that $\lambda_1 = \lambda_2 = 0$, $\lambda_3 = -¼$, and $\lambda_4 = ½$ generate point $\mathbf{y}$, which is therefore a linear combination of the other points. However, $\lambda_1 = 0$, $\lambda_2 = ½$, $\lambda_3 = ¼$, and $\lambda_4 = 0$ also generate point $\mathbf{y}$ which makes it a nonnegative linear combination as well. In order to find out whether or not $\mathbf{y}$ is an affine linear combination or even a linear convex combination of $\mathbf{x}^1$, $\mathbf{x}^2$, $\mathbf{x}^3$, and $\mathbf{x}^4$, we have to find a set of solutions to the system

$$\begin{aligned} 0\lambda_1 + 3\lambda_2 + 0\lambda_3 + 3\lambda_4 &= 1½ \\ 0\lambda_1 + 0\lambda_2 + 2\lambda_3 + 2\lambda_4 &= ½ \\ \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 &= 1 \end{aligned}$$

Ignoring the nonnegativity conditions for the time being (which are required for linear convex combinations), we apply the Gauss-Jordan pivoting algorithm. The table below shows the details of the calculations with the pivots, as usual, shown by the circled elements.

λ_1	λ_2	λ_3	λ_4	1
0	3	0	(3)	1½
0	0	2	2	½
1	1	1	1	1
0	1	0	1	½
0	–2	(2)	0	–½
1	0	1	0	½
0	1	0	1	½
0	–1	1	0	–¼
1	1	0	0	¾

One of the solutions to this system is $\lambda_1 = ¾$, $\lambda_2 = 0$, $\lambda_3 = ¼$, and $\lambda_4 = 0$, indicating that **y** is indeed an affine linear combination of $\mathbf{x}^1$, $\mathbf{x}^2$, $\mathbf{x}^3$, and $\mathbf{x}^4$. Finally, for **y** to be a linear convex combination of the given points $\mathbf{x}^1$, $\mathbf{x}^2$, $\mathbf{x}^3$, and $\mathbf{x}^4$, the following conditions must be satisfied: $\lambda_1 = ¾ - \lambda_2 \geq 0$, $\lambda_3 = -¼ + \lambda_2 \geq 0$, and $\lambda_4 = ½ - \lambda_2 \geq 0$, implying that $\lambda_2 \leq ¾$, $\lambda_2 \geq ¼$, and $\lambda_2 \leq ½$. It follows that for any $\lambda_2 \in [¼, ½]$, the resulting multipliers λ_1, λ_3, $\lambda_4 \geq 0$ and **y** is therefore also a linear convex combination of the given points.

As another example, consider the same given points $\mathbf{x}^1$, $\mathbf{x}^2$, $\mathbf{x}^3$, and $\mathbf{x}^4$ and let $\mathbf{z} = (1, 3)$ be an additional point. The reader will then be able to verify that **z** is both a nonnegative and an affine linear combination of the given points, but it is not a linear convex combination of these points.

A.3 Convexity

Definition A.28: A set *S* is said to be *convex*, if the linear combination of any two elements of *S* is also an element of *S*, i.e., if $\mathbf{x}, \mathbf{y} \in S$ and $\lambda \in [0, 1]$, then $\lambda\mathbf{x} + (1-\lambda)\mathbf{y} \in S$. Geometrically speaking, a set is convex, if all points on a straight line segment that joins any pair of arbitrary elements of *S* are also elements of *S*.

Figures A.3a – A.3f are examples for convex and nonconvex sets, where the shaded areas denote the sets defined by the relations shown below the respective figures.

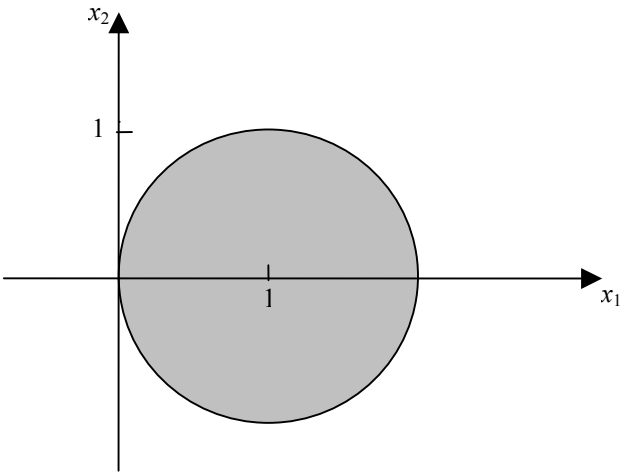


Figure A.3a

Figure A.3a shows a disk that is described by the inequality $x_1^2 + x_2^2 - 2x_1 \leq 0$. It is apparent that the set is convex.

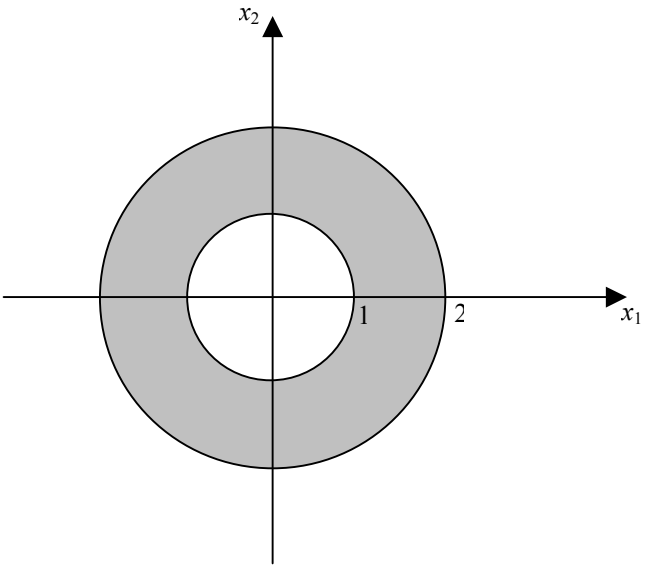


Figure A.3b

The shaded set in Figure A.3b is the difference of two disks. The inequalities that describe the set are $x_1^2 + x_2^2 \le 4$ and $x_1^2 + x_2^2 \ge 1$. The set is not convex.

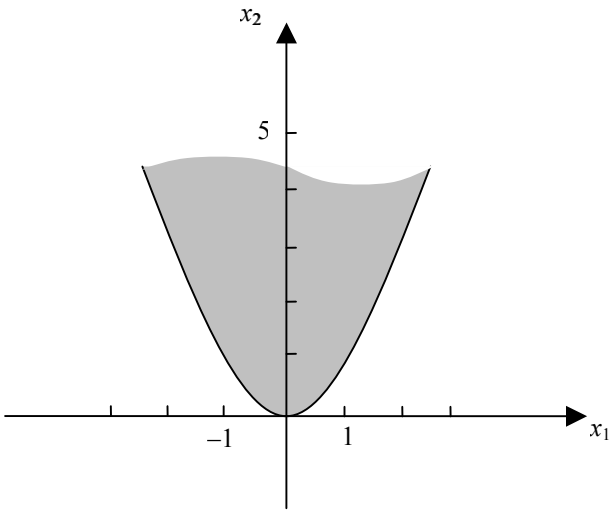


Figure A.3c

The area above in parabola in Figure A3c is described by the inequality $x_1^2 - x_2 \le 0$. It is a convex set.

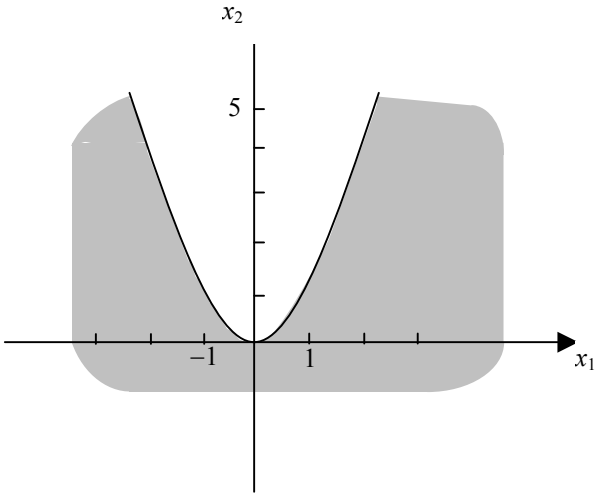


Figure A.3d

The shaded set in Figure A.3d is the complement of the set of Figure A.3c. It is described by the nonlinear inequality $x_1^2 - x_2 \geq 0$ and it is not convex.

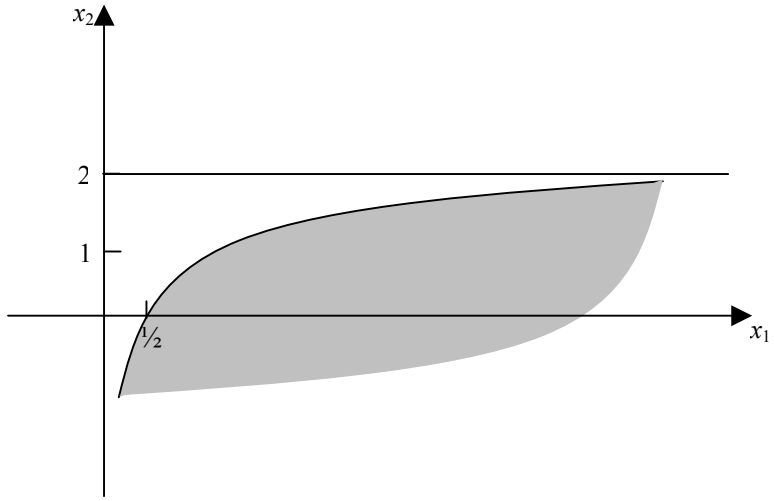


Figure A.3e

The shaded area in Figure A.3e is the area below the fractional function $\frac{1}{x_1} + x_2 \leq 2$, coupled with the nonnegativity constraint $x_1 \geq 0$. It is convex.

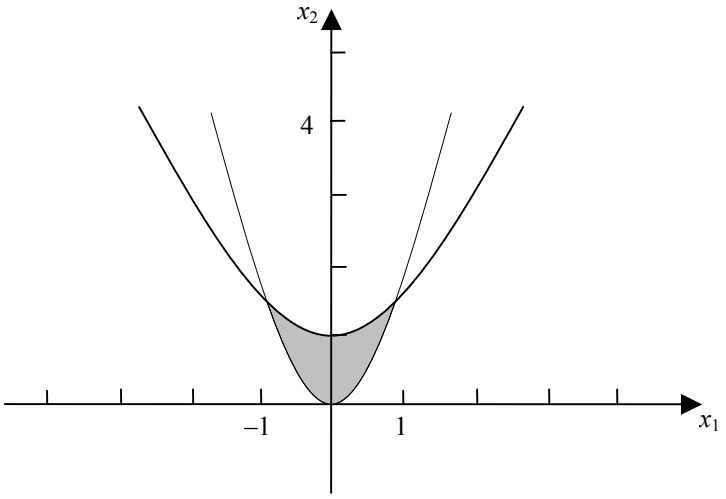


Figure A.3f

The set shown in Figure A.3f is the difference between two parabolas. The set satisfies the conditions $x_1^2 - x_2 \le 0$ and $-\frac{1}{2}x_1^2 + x_2 \le 1$, and it is not convex.

Other examples are $x_2 \le \sin x_1$ (not convex), $x_1^2 + x_2^2 > 0$ (not convex, since the origin in the (x_1, x_2) plane is not included in the set), and so forth.

The following example may provide an illustration of an algebraic proof of the convexity of a set.

Example: Consider the set given by $x_1^2 - x_2 \le 0$ as shown in Figure A.3c. Let $\tilde{\mathbf{x}}$ and $\hat{\mathbf{x}}$ denote any two points that satisfy the relation, i.e., $\tilde{x}_1^2 - \tilde{x}_2 \le 0$ and $\hat{x}_1^2 - \hat{x}_2 \le 0$. Let now $\bar{\mathbf{x}}$ be any linear convex combination of $\tilde{\mathbf{x}}$ and $\hat{\mathbf{x}}$, i.e., $\bar{x}_j = \lambda\tilde{x}_j + (1-\lambda)\hat{x}_j, j = 1,2.$

Then $\bar{x}_1^2 - \bar{x}_2 = [\lambda\tilde{x}_1 + (1-\lambda)\hat{x}_1]^2 - [\lambda\tilde{x}_2 + (1-\lambda)\hat{x}_2] =$

$$= \underbrace{\left(\hat{x}_1^2 - \hat{x}_2\right)}_{\le 0}\underbrace{(1-\lambda)}_{\le 0} + \underbrace{(\tilde{x}_1 - \hat{x}_1)^2}_{\ge 0}\underbrace{\lambda}_{\ge 0}\underbrace{(\lambda - 1)}_{\le 0} + \underbrace{\left(\tilde{x}_1^2 - \tilde{x}_2\right)}_{\le 0}\underbrace{\lambda}_{\ge 0} \le 0,$$

which proves that the set $\{x_1, x_2 \in \mathbb{R}: x_1^2 - x_2 \le 0\}$ is convex. Returning to linear relations, we can now state

Lemma A.29: Every linear relation defines a convex set.

Proof: Consider a linear relation $\mathbf{a}\mathbf{x}\ R\ b$ and suppose that $\mathbf{x}^1$ and $\mathbf{x}^2$ both solve this relation, i.e., $\mathbf{a}\mathbf{x}^1\ -\ b\ R\ 0$ and $\mathbf{a}\mathbf{x}^2\ -\ b\ R\ 0$. Let now $\bar{\mathbf{x}}$ be a linear convex combination of $\mathbf{x}^1$ and $\mathbf{x}^2$, i.e., $\bar{\mathbf{x}} = \lambda\mathbf{x}^1 + (1-\lambda)\mathbf{x}^2$, then

$$\mathbf{a}\bar{\mathbf{x}} - b = \mathbf{a}[\lambda\mathbf{x}^1 + (1-\lambda)\mathbf{x}^2] - b\ =$$

$$\underbrace{\lambda}_{\ge 0}\underbrace{(\mathbf{a}\mathbf{x}^1 - b)}_{R\ 0} + \underbrace{(1-\lambda)}_{\ge 0}\underbrace{(\mathbf{a}\mathbf{x}^2 - b)}_{R\ 0}\ R\ 0 \text{ for } R \in \{\le, +, \ge\}.$$

A similar argument applies to the cases in which $R \in <, >\}$. □

Now that we have proved that hyperplanes and halfspaces are convex sets, we can also prove the following

Lemma A.30: The intersection of a finite number of convex sets is a convex set.

Proof: First consider two sets. Let A and B be two convex sets and define $C = A \cap B$. For any two points $\mathbf{x}$ and $\mathbf{y}$ with $\mathbf{x}, \mathbf{y} \in A \cap B = C$, we can conclude that since

C is a subset of A as well as of B, $\mathbf{x}$ and $\mathbf{y}$ are both in A as well as in B. Define now a point $\mathbf{z} = \lambda\mathbf{x} + (1-\lambda)\mathbf{y}$, since the set A is convex by assumption, $\mathbf{z} \in A$, and since the set B is convex, $\mathbf{z} \in B$ follows, and hence $\mathbf{z} \in A \cap B = C$. Repeating this procedure, this proves the lemma for any finite number of sets. □

Since each polytope is, by definition, the intersection of hyperplanes and/or halfspaces, we can conclude that

Corollary A.31: A polytope is a convex set.

By definition, the empty set as well as a single point are also convex sets. With respect to linear relations, we will also need the following

Definition A.32: A *basic point* in $\mathbb{R}^n$ is the intersection of at least n hyperplanes at one single point; a *basic feasible point* is a basic point that satisfies all given linear relations. A basic feasible point is also called an *extreme point* of the set described by the linear relations.

As an illustration, consider Figure A.4.

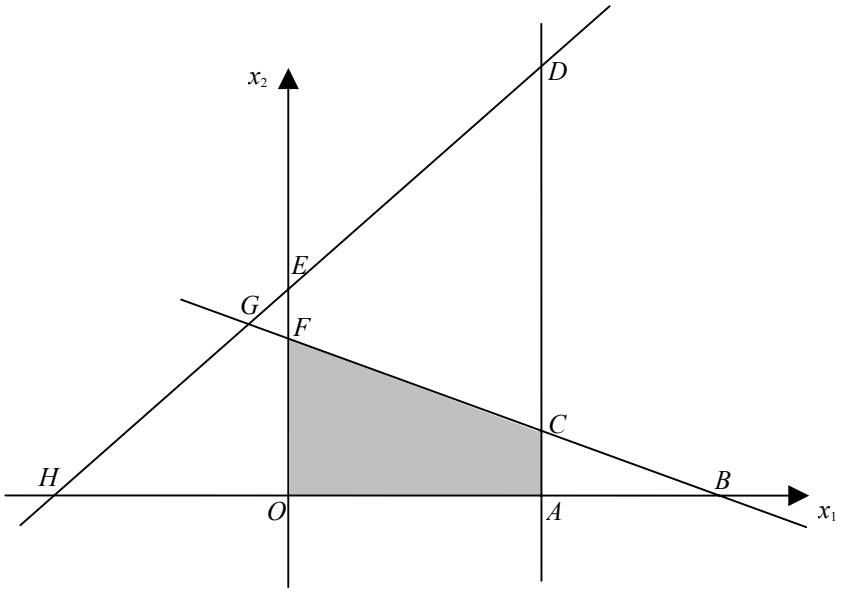


Figure A.4

The shaded area in Figure A.4 is the polytope; the points *0*, *A*, *B*, *C*, *D*, *E*, *F*, *G*, and *H* are basic points, whereas only *0*, *A*, *C*, and *F* are extreme points of the polytope.

We will conclude this section by stating

Lemma A.33: An extreme point cannot be expressed as a linear convex combination of other points in a polytope. In a polyhedron, each point can be expressed as linear convex combinations of extreme points. This property does not hold for polytopes that are not polyhedra.

A few examples may explain the ramifications of Lemma A.33. Consider a two-dimensional space. The only polytopes that have no extreme points at all consist of either the empty set, a single hyperplane or halfspace, or the intersection of any number of parallel halfspaces, whose gradients belong to a single hyperplane. The last object may be difficult to visualize; as an example, one could imagine an infinitely long pencil with flat surfaces. Polytopes with a single extreme point are either a point or a polyhedral cone as defined in Definition A.35 below. None of these polytopes are polyhedra and in none of these cases will it be possible to generate any point from the existing extreme points (if any) other than the extreme point itself.

Definition A.34: The set of all points that can be expressed as linear convex combinations of extreme points is called the *convex hull* of the given extreme points.

One can then prove

Proposition A.35: A polyhedron is the convex hull of its extreme points.

Definition A.36: A *convex polyhedral cone* is the intersection of any number of closed halfspaces whose bordering hyperplanes intersect at a single point.

Hence, each convex polyhedral cone is an unbounded polytope with just one extreme point, called its *vertex*. One can show that each convex polyhedral cone with its vertex at the origin can be generated as the set of all nonnegative linear combinations of a finite number of given points.

Definition A.37: A *simplex* $S \in \mathbb{R}^n$ is the convex hull of $(n + 1)$ given points, such that there exists no hyperplane in $\mathbb{R}^n$ that includes all of these $(n + 1)$ points.

As an example, a simplex in $\mathbb{R}^2$ is a triangle, and in $\mathbb{R}^3$ it is a tetrahedron. Figure A.5 shows simplices in 0-, 1-, 2-, and 3-dimensional real space.

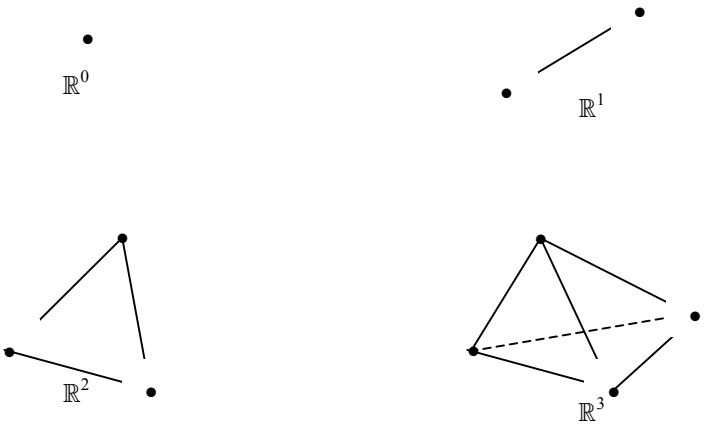


Figure A.5

B COMPUTATIONAL COMPLEXITY

B.1 Algorithms and Time Complexity Functions

Solving a mathematical problem will require calculations. Whether such calculations are performed manually or on a computer, we will be interested in the effort it will take to perform these calculations. Since the days of the abacus, the slide rule, and the mechanical calculating machine, our ability to handle vast amounts of calculations has developed to an extraordinary extent. However, at the same time, the need for ever more complex and large-scale computations has soared, at times outperforming the increase of computational capability. It is therefore interesting to study the science of how much calculation effort our numerical problems require. This is the topic of this chapter. The treatment in this chapter is based on material from Eiselt and Sandblom (2000). For an original and comprehensive treatment of computational complexity, readers are referred to the classical book by Garey and Johnson (1979).

In order to avoid confusion, we will use the term *problem* to denote a general mathematical formulation such as a linear, nonlinear, integer etc., programming model, whereas an *instance* of a problem will be any specific numerical case or realization of this problem; the size of the instance is typically expressed in terms of the number of variables, constraints, criteria, or similar measures.

Solutions can appear in two different guises: there are *closed-form solutions* and solutions that are not available in closed form. Simply speaking, closed-form solutions are formulas into which known numbers are inserted and the unknowns are then calculated. For instance, a quadratic equation $ax^2 + bx + c = 0$ has the closed-form solution $x = -\frac{b}{2a} \pm \sqrt{\frac{b}{4a^2} - \frac{c}{a}}$. Knowing the numbers a, b, and c, we simply insert them into the formula and solve for x. Clearly, most problems are much more complicated than that and they do not have closed-form solutions. In those cases, we typically have the need for a method that starts with some initial trial or estimated solution, from which it iteratively creates a succession of solutions whose quality improves as the method progresses. Such a technique is

called an iterative *algorithm* (named after the Arab mathematician Mohammed ibn-Musa al-Khwarizmi in the 9th century). For a detailed account, see the website by Overbay *et al*. At each iteration, the current solution is checked for a desired set of properties Π, such as optimality or feasibility. If the current solution satisfies the properties in Π, then the procedure stops; otherwise a new and improved solution is generated and the procedure is repeated. In order to formalize matters, let k be an iteration counter and denote by $\mathbf{x}^1$ the initial solution. We will write $\mathbf{x}^k \in \Pi$ if the solution $\mathbf{x}^k$ satisfies the properties included in Π. Finally, let $\tau(\mathbf{x}^k)$ be a transformation that generates a new solution $\mathbf{x}^{k+1}$ from the given solution $\mathbf{x}^k$. The basic structure of an iterative algorithm is outlined in Figure B.1.

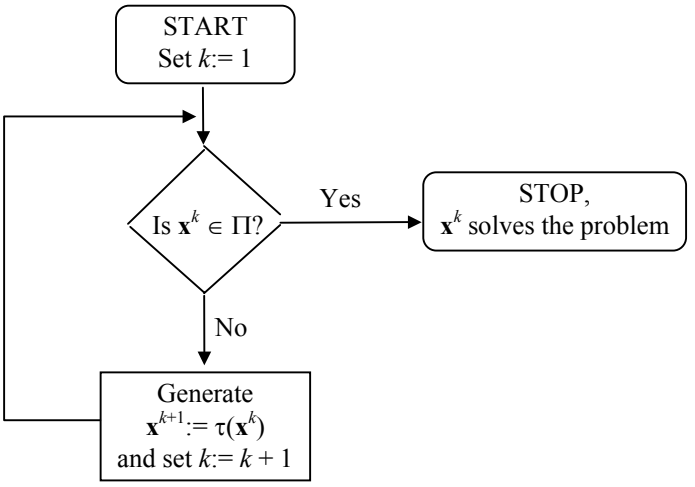


Figure B.1

As a simple example, consider bisection search, a technique that finds an item with a requested number in an ordered list, and apply it to the problem of finding a certain page in a book. For this purpose, it is immaterial whether or not there are some pages missing, but it is required that the pages (or, in general, documents) are in increasing or decreasing order. As an example, assume that the book has 480 pages and we would like to find page 293. Our search will always lead us from one guess to the next by searching between the low end (indicated by, say, a red bookmark) and the high end (indicated by a blue bookmark). In particular, the next guess will always lead us into the middle of an interval of uncertainty marked by its low and high ends. In our example, assume that our initial guess, i.e., the page we have presently opened, is page 120. As 293 > 120, we put the red bookmark on page 120 and insert the blue bookmark at the end of the book on page 480. Our next guess will then lead us to page 120 + ½(480 − 120) = 300. As

300 > 293, we keep the red bookmark at 120 and change the blue bookmark to page 300, knowing that the desired page will be located between the two bookmarks. The next guess is halfway between pages 120 and 300, i.e., at page 120 + ½(300 – 120) = 210. As 210 < 293, the red bookmark is moved from page 120 to page 210 and the search continues. The next search points are at pages 255, 278 (rounded up), 289 (rounded up), 295 (rounded up), 292, 294, and 293. At this point the algorithm has converged and the problems is solved.

In formal terms, $\mathbf{x}^1$ would be the initial page 120, the property Π would be "the page number is 293," and the iteration formula is the bisection search as described above.

Some algorithms do not need an initial solution to start with, as is the case with the Gauss-Jordan pivoting method which was described as Procedure A.19 in Chapter A.2 of this volume. The stop criterion was then Π: $I = \varnothing = J$, and the algorithm could stop with either a solution $\mathbf{x}^k$ in Step 4, or with the message in Step 2, indicating that no feasible solution exists. In general, stop criteria are typically associated with the two major properties of solutions to mathematical optimization problems: feasibility and optimality. These issues will frequently appear in the remainder of this book.

In principle, after an algorithm is designed, two properties will have to be verified, *viz*.,

- the *validity* of the procedure, and
- its *convergence*.

As far as the validity is concerned, we must prove that the solution produced by the procedure satisfies the properties in Π, and that, if such a solution does exist, it is found by the algorithm. The proof of convergence for exact methods is usually a proof of finiteness of the procedure, something that may not be required for certain approximation algorithms. As an example, consider the Newton-Raphson method (Procedure A.10 in Section A.2) for solving the equation $f(x) = 0$. Applying this algorithm to the function $f(x) = x^{1/3}$, we find $x_{k+1} = -2x_k$, which will not converge for any initial solution $x_1 \neq 0$. On the other hand, with $f(x) = x^2 - 2$, the procedure will converge to the solution $x = \sqrt{2}$, no matter what the initial solution x_1 is. However, this convergence is not finite, so that the stopping criterion for a finite conclusion of the algorithm needs to include one or more of the requirements $|x_{k+1} - x_k| \leq \varepsilon_1$, $|f(x_{k+1}) - f(x_k)| \leq \varepsilon_2$, and $k \geq M$, for some preselected (small) values of ε_1 and ε_2 and (large) value of M.

Once validity and finiteness are proved, the method can be implemented and applied. Over the years, however, scientists and practitioners alike have found that requiring finiteness of a procedure is not sufficient. For example, what good does it do if one can prove that an algorithm is finite if, even when applied to relatively

small models, it would take hours, days or even years of time to solve a given instance of the problem on a computer? This argument may seem far-fetched at first sight, but we will see in this chapter that problems of this type do indeed exist. Consequently, we need a better criterion than finite convergence and, in general, parameters that evaluate or classify an algorithm with respect to its performance.

Since all real world models will be solved on computers, we will focus on the requirements of algorithms as they are implemented on a computer. Typically, two criteria are used as performance criteria for algorithms: the amount of time required for the solution of a given model and the storage space requirements in the solution process. Clearly, the time requirement does not exclusively depend on the algorithm and the model itself but also on the specific computational equipment used, just as the required storage space for a model also depends on the way in which the model is coded. Since it is rather obvious that there is no general and definite answer to questions like: "How long will it take to solve any linear programming problem with the primal simplex algorithm?" even if the type of computational equipment were specified, the analyst may rephrase the question and either ask: "What is the expected time (or storage space) requirement?" or: "How much time (or storage space) will be required in the worst case?" if a specific problem is to be solved. The various types of analyses of an algorithm can be visualized in Figure B.2.

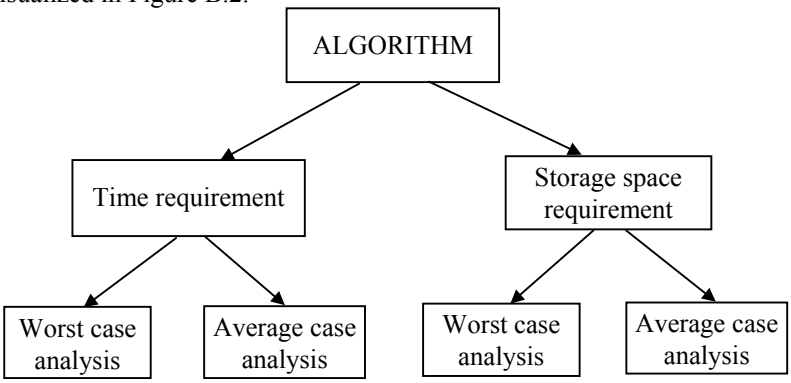


Figure B.2

In order to avoid discussing many details concerning the implementation of a model on a computer, we will restrict ourselves mostly to the worst case time analysis of an algorithm. In general, while it is generally possible to develop a theoretical upper bound for the time requirement in the worst case, an analysis of the average case either requires a probabilistic analysis using some statistical distribution for the numerical values of the problem coefficients, or an empirical study that involves solving a test series of either collected practical or random-generated instances of a problem. Examples for worst-case as well as probabilistic analyses are provided below.

In order to calculate the time requirements, it is usually beneficial to decompose the term "time" as

$$[\text{time}] = \left[\frac{\text{time}}{\text{elementary operations}}\right]\left[\frac{\text{elementary operations}}{\text{iterations}}\right][\text{iterations}]$$

The above term "elementary operation" is used for addition, subtraction, multiplication, division and comparison (i.e., finding the maximum or minimum of any two given numbers), and it is assumed that all elementary operations require the same constant time on a given computer, regardless of the magnitude of the numbers involved. This is obviously a simplification since a multiplication takes considerably longer than an addition or a comparison but this simplification does not change the principle of the argument. The time requirement for one elementary operation clearly depends on the specific computer model; currently it lies in the vicinity of a nanosecond, i.e., a modern computer is able to perform about a billion elementary operations per second. As the time requirement per iteration has nothing to do with the algorithm but rather with the state-of-the-art technology, we will just count the operations required for an algorithm and the number of iterations. If need be, the speed of the computer will be specified on an ad hoc basis geared towards the specific argument being pursued.

The last two factors in the above time function do not represent any fixed numbers that can be associated with a specific algorithm. They are functions depending on the size of the model or, in the above terms, the *size of the instance of the problem*. Since it is assumed that a digital computer is used for the solution of the model, the length of a *binary encoding* of the given instance is a reasonable measure of the size of the model. This yardstick does, however, sometimes prove to be rather awkward, so that we will usually resort to other "natural" measurements of size. As an example, consider a system of simultaneous linear equations. Here, the number of variables provides a reasonable description of the size of the model. If a string of n numbers is given that are to be sorted or otherwise manipulated, n is again used as a description of the size of the instance of the problem.

On the other hand, the magnitude of the numbers in the model is *not* a reasonable measurement of the size, at least not as far as digital computers are concerned. Such machines use binary encodings in which any given number b, e.g., the largest number in the model, is represented by $\log_2 b$ rather than b digits; for an introduction, see Lenstra and Rinnooy Kan (1979). Often, the number of elementary operations per iteration is easy to determine whereas the required number of iterations is either more difficult to compute or its only known bound may be exceedingly large.

Suppose now that the size of any instance of a problem can be described by n, and let $f(n)$ denote the maximal number of elementary operations required to solve any instance of the problem using the algorithm concerned. Then the function $f(n)$ is called the *time complexity function* of the algorithm. As an example, let the time complexity function of some algorithm be $f(n) = ½n^5 + 20n^4 + 2{,}000\ n$. We are specifically interested in the behavior of an algorithm applied to large-scale problems, i.e., problems with $n >> 0$, and observe that the term $½n^5$ then dominates the other terms in the sense that $f(n)/n^5$ tends to ½, the coefficient for the n^5 term, when n increases to infinity. Therefore an algorithm with the above time complexity function will be called of order n^5 or $O(n^5)$ for short. Loosely speaking, $O(f(n))$ includes only the fastest growing term of $f(n)$. As will be seen below, $O(f(n))$ is frequently much easier to determine than $f(n)$ itself.

We will call any algorithm for which $f(n)$ is bounded from above by a polynomial function in n a *polynomially bounded algorithm* or simply *polynomial algorithm*, whereas all other algorithms are termed nonpolynomial. Some authors use the *exponentially bounded algorithms* as a synonym for nonpolynomial algorithms, in which case the term "exponential" is used in the wide sense, regardless whether $f(n)$ is an exponential function in the strong sense or not. Edmonds (1965) was the first to use the term "good" for polynomially bounded algorithms. The reason to consider polynomial algorithms good and nonpolynomial algorithms poor performers (in the worst case) becomes clear considering Table B.1.

Table B.1

$O(\bullet)$	Time to solve a problem of size $n = 10$	$n = 20$	$n = 50$	$n = 100$	n_{hour}	N
$O(n^2)$	$(1)10^{-7}$ sec	$(4)10^{-7}$ sec	$(2.5)10^{-6}$ sec	$(1)10^{-5}$ sec	$n \approx 1.9$ million	$31.6n$
$O(n^3)$	$(1)10^{-6}$ sec	$(8)10^{-6}$ sec	$(1.3)10^{-4}$ sec	$(1)10^{-3}$ sec	$n \approx$ 15,326	$10n$
$O(n^5)$	$(1)10^{-4}$ sec	$(3.2)10^{-3}$ sec	0.31 sec	10 sec	$n \approx 325$	$3.98n$
$O(2^n)$	$(1)10^{-6}$ sec	$(1)10^{-3}$ sec	13 days	$(4.0)10^{13}$ years	$n < 41$	$n + 9.97$
$O(3^n)$	$(5.9)10^{-5}$ sec	3.5 sec	22.8 million years	$(1.6)10^{31}$ years	$n = 26$	$n + 6.29$
$O(n!)$	$(3.61)10^{-3}$ sec	77 years	$(9.6)10^{47}$ years	$(3.0)10^{141}$ years	$n = 15$	$n + 3$ for $n = 10,20$ $n + 2$ for $n = 50,100$
$O(n^n)$	10 sec	$(3.3)10^9$ years	$(2.8)10^{68}$ years	$(3.2)10^{183}$ years	$n = 11$	$n + 2.0$ $n = 10$ $n + 1.7$ $n = 20$ $n + 1.4$ for $n = 50$ $n + 1.2$ $n = 100$

Here we assume that the given computer requires one nanosecond (10^{-9} seconds) to perform an elementary operation. Table B.1 shows the computation times for a variety of polynomial and nonpolynomial algorithms when applied to problems of various sizes, and also provides values of n_{hour}, defined as the largest problem size n that can be solved within one hour by the algorithm under consideration. Finally, N denotes the size of problems that can be solved in the same time as a problem of size n, given a computer that is 1,000 times as fast.

With the unacceptably long computation times below the bold line in Table B.1, it is clear why polynomial algorithms can be labeled "good" and why nonpolynomial algorithms must be considered inefficient.

So far, we have discussed complexity functions which are assigned to specific *algorithms*. It is also possible to associate a worst case complexity function with every given *problem*. The complexity of a problem is then simply the complexity function of the best known algorithm that solves the problem.

B.2 Examples of Time Complexity Functions

In this section we will provide some simple algorithms whose worst case complexity can easily be determined. The principle in selecting these algorithms has been that they should be simple and not anticipate concepts to be described later in this volume. This may make the exposition fairly elementary, but it should make the principle sufficiently clear. The methods are mostly chosen from the areas of linear algebra and file organization in data processing.

Example 1: Solution of systems of simultaneous linear equations.

Problem: Given a matrix $\mathbf{A} \in \mathbb{R}^n \times \mathbb{R}^n$ and a vector $\mathbf{b} \in \mathbb{R}^n$, find a solution $\mathbf{x}$ to the system $\mathbf{Ax} = \mathbf{b}$ by applying the Gauss-Jordan pivoting method, described as Procedure A.19.

Complexity: In each iteration the transformation of a single element takes constant time; given that there are n^2 elements in the matrix, each iteration has a complexity of $O(n^2)$ and the fact that n iterations are needed leads to an overall complexity of $O(n^3)$. Note that since each of the n^2 elements given in the problem has to be considered and used at least once in an elementary operation during any solution method, an obvious lower bound for the complexity function of the problem is $O(n^2)$.

Example 2: Evaluation of a determinant.
Problem: Given a matrix $\mathbf{A} \in \mathbb{R}^n \times \mathbb{R}^n$, evaluate the determinant of $\mathbf{A}$.

Complexity: We will apply two different methods to solve this problem. First consider Gauss-Jordan pivoting. In this method we pivot in each row and in each column no more than once, and then the product of the pivots is *det* **A**. As in Example 1, $O(n^2)$ elementary operations are required in each Gauss-Jordan iteration, and since no more than n iterations (and less if **A** is singular) are needed, we obtain a computational complexity of $O(n^3)$. In addition, the product of the pivots is calculated in $O(n)$ multiplications, yielding an overall complexity of $O(n^3) + O(n) = O(n^3)$.

The second method is the Laplace expansion, see Definition A.8. It evaluates the determinant by the i-th row, by computing

$$\det \mathbf{A} = \sum_{j=1}^{n} (-1)^{i+j} \det \mathbf{A}_{ij} ,$$

where $\mathbf{A}_{ij}$ denotes the submatrix of **A** obtained from **A** by deleting its i-th row and its j-th column. In the worst case $a_{ij} \neq 0 \; \forall \; j = 1, ..., n$ so that det **A** is now expressed as the sum of determinants of n matrices of dimension $[(n - 1) \times (n - 1)]$ each of which, in turn, can be expressed as the sum of determinants of $(n - 1)$ matrices of dimension $[(n - 2) \times (n - 2)]$ and so forth. The procedure continues until the determinant is expressed in terms of determinants of $[2 \times 2]$ matrices, which can be evaluated in three elementary operations each. The total number of elementary operations is roughly $3 \prod_{k=2}^{n} k$ which is $O(n!)$. Hence this technique has a nonpolynomial complexity and is therefore computationally inferior to the pivoting method above, at least as far as the worst case is concerned. However, this method is still quite instructional and useful for some proofs.

Example 3: Multiplication of two matrices.

Problem: Given two matrices **A**, $\mathbf{B} \in \mathbb{R}^n \times \mathbb{R}^n$, determine the product **C** = **AB**, using

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \quad \forall \; i, j = 1, ..., n.$$

Complexity: Each element c_{ij} is the inner (or scalar) product of the n-vector $\mathbf{a}_{i\bullet}$ and the n-vector $\mathbf{b}_{\bullet j}$. The calculation of each element c_{ij} thus requires n multiplications and $(n - 1)$ additions, i.e., $O(n)$ elementary operations. Since **C** includes n^2 elements, n^2 inner products have to be calculated resulting in a complexity of $O(n^3)$. Note that since each of the $2n^2$ elements of **A** and **B** is used in at least one operation, a lower bound of $O(n^2)$ exists for the complexity. The reader is referred to the decomposition principle of Strassen (1969) with improved complexity for this problem.

Example 4: Search methods.

Problem: Given an ordered file with n numbers a_j, $j = 1, ..., n$ such that $a_1 \le a_2 \le ... \le a_n$ and each number is allocated the same amount of space on the file. Given some number a^* (the search argument), find j, such that $a_j = a^*$.

Complexity: The *Sequential Scanning Method* compares a^* with a_1, if $a^* = a_1$, stop; otherwise compare a^* with a_2 and so forth. Clearly, in the worst case $a^* = a_n$ and n comparisons are necessary; hence the complexity is $O(n)$. Let $P(a_j)$ now denote the probability that $a^* = a_j$. If $P(a_j) = 1/n \; \forall \, j$, then the expected number of comparisons (the average case analysis) is

$$\sum_{j=1}^{n} P(a_j)\, j = (n+1)/2$$

which is still $O(n)$. However, if numbers that are frequently requested are placed at the beginning of the file, the average time savings can be significant.

For the so-called *Bisection Search Method* that was illustrated in the book example at the beginning of Section B.1, suppose that $n = 2^k$ for some $k \in \mathbb{N}_0$. Compare a^* with $a_{n/2}$; if $a^* \le a_{n/2}$ then discard all a_j with $j > n/2$ and if $a^* > a_{n/2}$, discard all a_j with $j \le n/2$. Then the process is repeated in the remaining interval. It can be seen that in each iteration the relevant file length is cut in half; since only one comparison is required per iteration, it follows that the complexity is of order $O(\log_2 n)$. As an illustration of the above, let $n = 16$ and suppose that $a^* = a_{11}$. Then the arrows in Figure B.3 show the steps in the bisection search procedure.

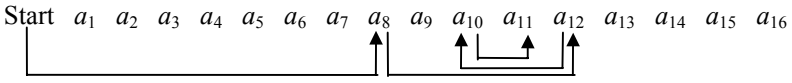


Figure B.3

The bisection search procedure is extremely efficient; even files with one billion numbers require no more than thirty comparisons. It should also be mentioned that with $P(a_j) = 1/n \; \forall \, j$, the average case complexity is almost identical to the worst case complexity.

Using the *Matrix Search Method*, suppose that the numbers a_j, $j = 1, ..., n$ are arranged in a $\left[\sqrt{n} \times \sqrt{n}\right]$-dimensional matrix so that $a_{11} \le a_{21} \le ... \le a_{m1} \le a_{12} \le a_{22} \le ... \le a_{m2} \le a_{13} \le ...$ where $m = \sqrt{n}$, i.e., the ordered file is broken down into $\sqrt{n}$ -dimensional subfiles which are the columns of the matrix. The procedure is to scan the elements $a_{11}, a_{12}, ..., a_{1\sqrt{n}}$ (i.e., the first row of the matrix) until for some k, $a^* > a_{1k}$ for the first time or $a_{1\sqrt{n}}$ has been reached, in which case we set

k:=1+$\sqrt{n}$. Then sequential scanning continues in the (k – 1)-st column. In the worst case all $\sqrt{n}$ elements in the first row and the $\sqrt{n}$ elements in column (k – 1) have to be scanned yielding a worst case complexity of $2\sqrt{n} = O\left(\sqrt{n}\right)$, which equals the average complexity. A possible improvement is to conduct a binary, rather than sequential, search in the row and later the column of the matrix.

Search in an unordered set. Although sequential scanning is by far inferior to the two other search methods described above, it is the only one which can also be used for a search in an unordered file. Its complexity does not change whether the file is ordered or not. In many applications, however, the probabilities $P(a_j)$ are not uniformly distributed. Consider, for example, a file of customer numbers. Each time a customer places an order, its number has to be sought, and from there a pointer leads to a customer's file, e.g., his address, most recent orders, etc. Clearly, there will be customers who order frequently, as well as customers whose orders come in infrequently. Since the numbers of "good" customers have to be sought much more frequently, it seems sensible enough to order the file, such that $P(a_1) \geq P(a_2) \geq ... \geq P(a_n)$. This may dramatically reduce the average number of necessary comparisons, although it does not change the worst case complexity, as the following example may illustrate.

Let the customers of a company be divided into three groups in such a way that group one comprises ten percent of the customers, group two consists of twenty percent of the customers and group three includes the remaining seventy percent. On average, each customer in group one orders five times as often as any customer in group two who, in turn, places ten times as many orders as a customer in the third group. Assume that a_1, ..., $a_{n/10}$ are the customer numbers for group one; $a_{n/10+1}$, ..., $a_{3n/10}$ are the numbers for members of group two and the last (3n/10)-1 numbers are associated with customers from group three. Then the probability distribution is

$$P(a_j) = \begin{cases} 500/77n & \text{for } j \leq n/10 \\ 100/77n & \text{for } j \in]n/10; 3n/10] \\ 10/77n & \text{for } j > 3n/10 \end{cases}$$

The expected number of comparisons is then

$$\sum_{j=1}^{n/10} (500/77n)\, j + \sum_{j=n/10+1}^{3n/10} (100/77n)\, j + \sum_{j=3n/10+1}^{n} (10/77n)\, j = (221n - 770)/1{,}540$$

which approaches 0.1435n for large values of n. For n = 1,000 customers, rather than an average of 500.5 comparisons for a completely unordered file, only 143 comparisons are needed.

Example 5: Sorting methods.

Problem: Order a file of n numbers $a_1, a_2, ..., a_n$, so that $a_1 \le a_2 \le ... \le a_n$.

Complexity: The simplest technique is *Sorting by Selection*. Find the minimum among all elements on the file (requiring $O(n)$ comparisons), replace it by some very large number $M >> 0$, and write it on a separate file. After repeating this procedure n times, the new file includes all numbers in nondecreasing sequence as desired, whereas the original file consists of only M's. Hence the overall complexity is $O(n^2)$.

The *Pairwise Exchange Method* first compares the first and second, the third and fourth, the fifth and sixth, etc. numbers with each other and exchange them if they are not in nondecreasing order. In the next step, compare the second with the third, the fourth with the fifth, the sixth with the seventh element, etc. and rearrange again if necessary. All odd-numbered iterations are identical to the first, and all even-numbered iterations are the same as the second iteration. The procedure terminates as soon as no two numbers are exchanged in two successive iterations. In each iteration, $\frac{1}{2}n$ comparisons are required, and no more than $O(n)$ iterations are necessary. This becomes intuitively clear by considering the worst case in which the smallest number which should appear in the first position on the file is currently at the very end of the file. In each iteration, this number moves one step towards the beginning of the file. This yields an overall complexity of $O(n^2)$ of the procedure. As an illustration, consider the following example in Figure B.4 where the brackets indicate the pairs of numbers that are considered for a possible change, and an asterisk next to the bracket shows that a pairwise exchange was made.

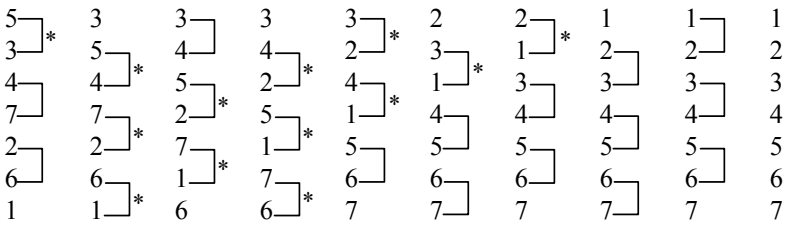


Figure B.4

B.3 Classes of Problems and Their Relations

In this section we will define classes of problems and explore some of the relations between them. In order to do so, it is necessary to rephrase the known *optimization problems* as *recognition* or *decision problems* for which the result is

a "yes" or "no" answer to the question: Is **x*** a solution to the instance of a problem? Our treatment here only covers material needed for the rest of this book. For a comprehensive account, see Garey and Johnson (1979).

Given the complexity function $O(f(n))$ defined in the previous section, we write

Definition B.1: The class **P** is the class of problems for which there exists an algorithm with a complexity function $O(f(n))$ that is polynomial in the problem's input parameter(s).

Similarly, we can write

Definiton B.2: The class **NP** is the class of decision problems for which a guessed solution can be verified in polynomial time.

As an example, consider the problem of finding a feasible solution to a system of simultaneous linear inequalities. Any guessed solution could be verified in polynomial time, thus the problem is in **NP**.

The above definition immediately leads to

Lemma B.3: P $\subseteq$ **NP**.

A very important unsolved problem is whether the above inclusion is a proper one or not.

Definition B.4 A problem $M \in$ **NP** is called **NP**-complete, if each instance of every problem in **NP** can be transformed into an instance of M in polynomial time. The class of all **NP**-complete problems is called **NPC**.

It follows from Definition B.4 that an **NP**-complete problem is at least as difficult to solve as any problem in **NP**.

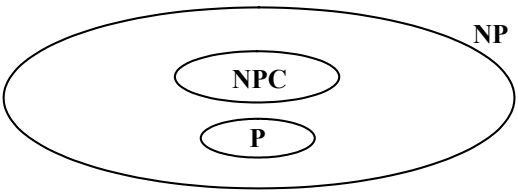


Figure B.5

If any decision problem is in **NPC**, then the corresponding optimization problem is **NP**-*hard* because the existence of a polynomial algorithm for the optimization problem would imply that the decision problem can also be solved in polynomial time. The relations between **P, NP**, and **NPC** can be visualized in Figure B.5 (assuming that $\mathbf{P} \neq \mathbf{NP}$).

As already mentioned above, the fundamental question today is the relation between **P** and **NP**, i.e., $\mathbf{P} = \mathbf{NP}$ or $\mathbf{P} \underset{\neq}{\subset} \mathbf{NP}$? It is believed that **P** is a proper subset of **NP**, but it seems that the existing mathematical tools are not sufficient to prove or disprove this claim. Since it has been shown that all problems in **NPC** are polynomially equivalent, the existence of a polynomial algorithm for any problem in **NPC** would imply that $\mathbf{P} = \mathbf{NP}$. On the other hand, Ladner (1977) has shown that $\mathbf{P} \neq \mathbf{NP}$ implies that there exists at least one problem M, such that $M \notin \mathbf{P}$ *and* $M \notin \mathbf{NPC}$, i.e.: $\mathbf{NP} \underset{\neq}{\supset} \mathbf{P} \cup \mathbf{NPC}$.

In general, one can distinguish between four classes of problems.

- *Easy problems*, i.e., problems for which polynomially bounded algorithms exist. Problems in this category are in class *P* and examples of problems in this class are provided in the previous section.

- *Probably difficult problems*, i.e., problems for which at this point only exponentially bounded algorithms exist, but it is not known whether or not polynomial algorithms exist for these problems. This is the class **NPC**.

- *Intractable problems* are provably difficult problems, i.e., problems for which proof exists that no polynomial algorithms can possibly exist for any problem in this class.

- *Undecidable problems*, i.e., problems for which we have proof that no algorithm can possibly exist to solve them. Examples for problems in this class are the question whether or not a computer comes to a halt after a finite number of steps for any given program and any given input string, the problem of finding integer solutions to a set of polynomial equations (Hilbert's tenth problem), or nonlinear integer programming, see Jeroslow (1973).

Finally a word with respect to the average and worst-case time analysis discussed earlier in this chapter. It was shown at the beginning of this chapter that a good worst case time complexity is a crucial feature of a successful algorithm. Consider, however, any of the simplex algorithms on the one hand and the ellipsoid method on the other. Both methods solve linear programming problems and are described in detail in Chapters 3 and 7, respectively. With respect to worst case time analysis, the simplex methods are poor performers whereas the ellipsoid method performs well. Still, most linear programming problems in practice are

solved by one of the simplex methods because their average performance is much better than that of the other approaches for linear programming problems. Moreover, the simplex methods have a distinct advantage over the other technique as far as storage space requirement is concerned. But even a combination of all four criteria, worst case, and average case analysis in terms of time and storage space requirement is not sufficient to evaluate an algorithm. It must also be possible to easily take advantage of special structures (such as block-angular or staircase structures or transportation and assignment-type structures in the simplex methods) and, very important, the method should allow the user to perform sensitivity analyses efficiently, a factor which certainly is one of the deciding factors that has contributed to the success of the simplex methods.

1 INTRODUCTION

This chapter will first outline some of the major developments that have led to the field that is now known as linear programming. We will then summarize the structure and the main assumptions made in linear programming. The third section describes the modeling process in some detail. Section 4 discusses the main three phases in optimization, and the last section in this chapter solves a small linear programming problem and interprets a computer printout.

1.1 A Short History of Linear Programming

The subject of this book, Linear Programming, is a subset of mathematical programming which, in turn, is a part of operations (or operational) research. Operations research, also referred to as management science, is a discipline that deals with the optimization and control of systems. The term "programming" is used here as a synonym for optimization. Interestingly enough, optimization problems are nothing but formalized versions of the fundamental economic principle: they either maximize the output for some given input, or minimize the input for some required output. Which of the two versions applies to any given problem depends on the specific scenario or the viewpoint of the decision maker.

The history of linear programming can be traced back to the 1930s and 1940s. The earlier part of the history is described by McCloskey (1987), while a recent authoritative account is provided by Gass and Assad (2005). Here, we provide only a few of the highlights that have directly impacted the field of linear programming. An interesting collection of personal reminiscences is found in Lenstra *et al*. (1991), including Dantzig's (1991) contribution; for an earlier account see also Dantzig (1982).

After early precursors that mostly dealt with linear (in-) equalities by Fourier (1826) (for his contribution, see also Grattan-Guinness (1970)), Gauss (1826), Gordan (1873) and Pareto (1906) as well as contributions in game theory (a part of which was later shown to be closely related to linear programming), more

closely related work on the subject began in the mid-1930s. Motzkin's (1936) solutions of systems of linear inequalities, Leontief's (1936) work on input-output models, Kantorovich's (1939) production assignments, and Hitchcock's (1941) transportation problem are the main contributions that dealt with still isolated attempts to quantify and solve practical problems that could be reduced to linear systems.

While operations research and linear programming had not been formally described, the theory of games had already reached a peak with the publication of von Neumann and Morgenstern's (1944) tome "The Theory of Games and Economic Behavior." It was in 1947 that Dantzig first devised an automated technique that allowed problems with linear constraints and a linear objective to be solved—in theory, at least. The explosive development of operations research after that description is in large part due to advances in computational technology. Among the first computer implementations of the simplex method is a model with 71 variables and 48 constraints that took no less than 18 hours to solve. This may be a small problem by today's standards, something that any pertinent software could solve within a fraction of a second, but at that time, it was a breakthrough that, more or less for the first time, allowed a practical problem to be solved with Dantzig's newly developed method.

Dantzig's findings were presented at a conference organized by the Cowles commission in 1949 and the proceedings of this conference were published two years later in a book edited by Koopmans (1951). Already in 1950 the first operations research journal, the British "Operational Research Quarterly," was put out. Two years later the journal "Operations Research," the publication of the Operations Research Society of America, followed.

After the initial groundwork was laid, improvements to existing methods were made, new mathematical structures were found useful and were described, and new fields were discovered and developed. In linear programming, Manne (1953) and Orchard-Hays (1955) dealt with parametric programming (the latter author's results were based on his unpublished Master's Thesis three years prior), while Orden (1952) described the product form of the inverse. Charnes (1952) dealt with an aberration of linear programming called degeneracy, and 1953 saw the emergence of the computationally efficient revised simplex method by Dantzig and Orchard-Hays (1953). The same year also saw one of the first texts on linear programming written by Charnes *et al*. (1953). Other groundbreaking developments were the development of the dual simplex method by Lemke (1954), the design of cutting planes for integer programming problems by Dantzig *et al*. (1954) and the use of branch and bound by Land and Doig (1960), also for integer programming problems, seemingly straightforward extensions of standard linear programming problems.

Progress was, however, not confined to improvements of the existing tools. New applications were developed as well. Markowitz (1952) applied mathematical

(albeit nonlinear) programming to portfolio selection in finance, Charnes *et al*. (1952) dealt with an application of linear programming in the blending of aviation fuels, Ford and Fulkerson (1954) described a linear programming problem that would enable a planner to ship as many units as possible from an origin to a destination through a network, and Koopmans and Beckmann (1957) were the first to describe the quadratic assignment problem, again a nonlinear structure, that was found to be applicable to facility layout problems.

The 1960s saw more developments in many fields related to linear programming, particularly integer programming, nonlinear programming, and flow problems. In addition, computers were beginning to make their mark in the increase of problem sizes that could be solved. This was one of the driving forces of progress for linear programming models and applications as well as for operations research in general. Edmonds (1965) introduced the notion that algorithms that require a number of operations that is some polynomial function in the length of the input in the worst case should be considered efficient. This development anticipated complexity theory and the analysis of algorithms that became a major issue in the next decade.

In the 1970s, the issue of algorithmic efficiency began to be taken very seriously with the contributions by Cook (1971) and Karp (1972). Starting with automata theory, the first author described a provably difficult problem called "Satisfiability" or SAT for short, and the second author described a reduction theme, according to which problems may be reduced to Satisfiability. In case such a reduction exists, the problem under consideration is proved to be at least as difficult as SAT. This allowed scientists to classify some problems as difficult or hard, while others would be easy even in the worst case. Given that distinction, it was of special interest to researchers in and users of linear programming when Klee and Minty (1972) determined that at least one version of the simplex method was not efficient in the worst case.

Based on earlier work by Shor (1970), Khachian (1979) described an "ellipsoid method" that was able to solve linear programming problems efficiently in the worst case. The method turned out to be painfully slow in practice, so the achievement was largely theoretical. This changed with the work by Karmarkar (1984), whose "interior point methods" had the property that the computational effort they required would increase only marginally as the size of the problem increased. This makes them uniquely suited to large-scale practical problems. Work on improvements to methods in this class and the search for more efficient implementations of interior point methods continues to this day.

Other advances have been the proliferations of user-friendly software including the use of spreadsheets in optimization, and the hardware to go along with it, whose ever-increasing speed and memory and decreasing costs have greatly advanced the use of optimization techniques in general and linear programming in particular.

1.2 Assumptions and the Main Components of Linear Programming Problems

Mathematical programming problems are mathematical models that attempt to model a real-life situation. They do so by using *variables* and *parameters*. Both represent numbers, but while parameters are numbers that are known to the decision maker and have to be taken as a fixed datum, variables are numbers whose values will be determined in the process. In general, parameters are not within the jurisdiction of the decision maker, while variables are. This concept may be best explained by a small example. Suppose that an individual wants to plan his diet in a way, so that the nutritional content of the diet satisfies generally accepted standards, while the costs are minimized. (This is the famed diet problem which is formally modeled in the next chapter). Here, the content of nutrients in the foodstuffs under consideration are parameters, and so are the prices of the foodstuffs and the quantities of nutrients that should be included in the diet. On the other hand, the quantities of foodstuffs that are included in the diet are within the jurisdiction of the decision maker and hence they are decision variables, while the quantities of nutrients in the diet, while determined by the decision maker through the food intake, are not directly controlled by him. These are endogenous variables, but not decision variables in the narrow sense. In this book the distinction between parameters (sometimes called exogenous variables) and variables is sufficient.

Some decision makers prefer to further subdivide the variables into two classes based on causality. Consider a simple example: in a model of a national economy, the supply of money and the unemployment rate are both unknown, i.e., they are not parameters. However, while the supply of money can be determined directly by the federal government (or, more specifically, by the Federal Reserve), the rate of unemployment will be a result of the government's policy. By changing the variables under its jurisdiction, the national planners can influence the unemployment rate, but only indirectly by setting the variables under their jurisdiction to the desired values. This distinction is not relevant in the models under consideration here: all variables are included in the model, and the solver will determine their optimal values, regardless if they can be influenced directly by the decision maker or not. Actually, in order to distinguish the variables included in the model by the decision maker from those added by the solver in the solution process, we will call all these variables "decision variables," regardless if the decision maker can directly choose their value or not.

All mathematical programming problems consist of two components, *viz*., constraints and objectives. Constraints are imposed by the system, meaning they are not within the jurisdiction of the decision maker and all he can do is realize their existence and respect them. Typical examples are constraints that limit resources (e.g., budget constraints), existing contractual agreements that require that certain quantities of products are delivered, certain manpower assignments are not made (due to collective agreements), physical and/or chemical limits, etc.

It should be noted that constraints in mathematical programming cannot be violated. For instance, if only ten units of a resource such as machine time are available, then no schedule will be considered that uses more than ten units, regardless how many more units. This may very well be in conflict with reality: for instance, a budget constraint that limits weekly expenditures to, say, $1,000, may very well be violated in practice by taking out a loan.

Most beginners will be only too eager to formulate constraints, even if the restrictions are not at all hard. Some examples are provided by Eiselt and Laporte (1987). Among them are problems that schedule students' final exams. The softness of constraints can nicely be shown in such a context: having a student write two exams at the same time: impossible, i.e., a hard constraint; having a student write two exams in a row on the same day: highly undesirable, so a high penalty is added to such a case; having a student write two exams on the same day, one early in the morning and the other late in the afternoon: quite undesirable, so there is some penalty; having a student write his first exams very early in the exam period and his last exam very late in the exam period: somewhat undesirable, so that a small penalty is added to such a case. As can be seen from the above example, only in the case of an impossibility should a constraint be formulated that prohibits that particular instance; in all other cases of undesirability, penalties should be defined which are subsequently incorporated in the objective function rather than the constraints. For some pertinent comments, see also Moore and Weatherford (2001). Some authors have approached the modeling of soft constraints by using fuzzy programming. Some details are found in Chapter 9 of this volume.

On the other hand, there are objectives that express the wishes of the decision maker. Most optimization models employ a single objective function. Again, Chapter 9 in this volume examines models that have more than a single objective. As a matter of fact, whenever more than one objective is present, the concept of optimality—a key concept in optimization—loses its meaning and has to be replaced by other, conceptually weaker, concepts. One can think of an optimization as a pasture, whose boundaries are the fences (our constraints), while the objective function points into a direction, in which the grass is greener and we would like to go as far as possible, and that is exactly the function of the solver that is applied to the optimization problem.

We will now investigate the two problem components, constraints and objectives, in more detail. First consider the objective function. There are two ways an objective function can be written, *viz*., Max $z_f = f(x)$ or Min $z_g = g(x)$. To the solver it is completely irrelevant if the objective measures profit, revenue, sales, or some other utility. All such measures have in common that they maximize the objective. Similarly, it is irrelevant if our objective expresses costs, distances, or any other disutility, as all such measures are going to be minimized. As a matter of fact, since each maximization objective Max $z_f = f(x)$ can be written as an equivalent minimization objective Min $z_g = g(x)$ with $z_g = -z_f$ and $g(x) = -f(x)$, there is no

need to devise methods for both types of problems, since each minimization problem can easily be transformed into an equivalent maximization problem and vice versa. Also, fixed costs may or may not be included in the objective function as they do not influence the optimization.

What is by no means obvious or easy to state is the actual expression of the objective. Typically, the true objective of the decision maker will remain elusive. It is often a very general statement such as "improve the firm's efficiency," "satisfy our customers as best as possible to generate repeat business," "distribute our products as quickly as possible," or a similar statement that will have to be quantified. Typically, the analyst will determine a *proxy* or *surrogate criterion* that, at least hopefully, will measure roughly what the decision maker has in mind. The proxy objective, while it may not be exactly what the decision maker wants, will have the advantage of being quantifiable.

This process does sound easier than it normally is. "Distributing products quickly," for instance, can mean many different things. If there were just two customers and two solutions were possible: one that distributes the products to our two customers in three days each, and another that delivers the goods to customer one in a single day and to customer two in four days, which of the two solutions would the decision maker prefer? In other words, is the average or the longest delivery time the appropriate yardstick?

Problems such as these occur even in business situations, where normally most measures finally are reducible to the common denominator of dollars. This is true even in cases in which the decision maker wants to apply objectives that appear very difficult to quantify, e.g., the damage of an oil spill or the risk of a fatality in the airline industry. After all, it is always possible to take out insurance for such cases which will reduce these risks and environmental damages to costs. However, customer satisfaction and high levels of employee satisfaction, both long-term objectives that ensure repeat business and good labor relations, even though ultimately they will result in higher profits, remain elusive as far as their quantification is concerned.

Similarly, the simple objective "maximize profits" may be achieved by different means. In the short run, this objective may be achieved by increasing prices of some goods. The long term implications may, however, be stagnating growth, shrinking market shares and finally decreasing profits; the maximization of profits in the long run may be achieved by a completely different strategy. In the public sector things are still more complicated. "Maximize the quality of services" could be substituted by maximizing the number of civil servants; a highly questionable proxy expression, as pointed out by Parkinson (1957). For a hospital administration, the objective may be to maximize the quality of hospital service. An analyst attempting to quantify the real criterion "hospital service" may apply the proxy z = the average number of days a patient stays in a hospital. Minimizing z could imply discharging patients not completely cured, while maximizing z may

have the effect of the hospital not admitting new patients with serious diseases and in need of a hospital bed.

There are also other aspects that make decision making in the public sector more complicated than in the private sector. For one, there are typically multiple decision makers or stakeholders in public decision making scenarios, coupled with a large number of objectives. Worse yet, many of the model's parameters will not be agreed upon: the evaluation of risk will widely vary among those involved (typically the risk is considered much higher by those directly involved), leaving the door open to "junk science" and similar abuses.

Next, consider the constraints of a model. The structure of all constraints, regardless what model they belong to, is the same. A constraint can be written as

$$LHS\ R\ RHS,$$

where *LHS* denotes "left-hand side", *R* denotes a (mathematical) relationship, and *RHS* symbolizes the right-hand side of the relation. In particular, the left-hand side always measures the value associated with a real situation, the relation $R \in \{\leq, =, \geq\}$, and the right-hand side is a single number that provides the yardstick with which the value on the left-hand side is compared. In that sense, a constraint is not unlike a typical statistical hypothesis test, in which a test statistic is compared with a benchmark value. Based on the relationship between these two values, the hypothesis test is either accepted or rejected. As an example, consider a typical resource constraint, such as a budget constraint. On the left-hand side *LHS* we will formulate an expression for the amount of money that is actually spent, the right-hand side *RHS* will express how much money we are allowed to spend, and the relation *R* in this case will be "≤." Note that from a technical point of view, it is always possible to convert a ≤ constraint to a ≥ constraint and vice versa by multiplying the constraint by any negative number, so that it is not limiting when we consider only ≤ constraints here.

When formulating constraints, it is always beneficial to first formulate the requirement as a sentence of the English language, which then can be translated into a mathematical structure. It should be ensured that the analyst is able to interpret each single term of the expression. As an example, consider again a budget constraint. Let there be two products, whose unit prices are \$5 and \$8, respectively, and whose quantities we denote by x_1 and x_2, respectively. Assuming that \$60 are available to us during the planning period, we can then require that our actual expenditures should not exceed the amount of money that we have. The actual expenditure can be decomposed into two components, one that expresses the amount of money we spend on the first product and another that measure the amount that we spend on product two. Given that each unit of product one costs \$5 and we purchase a total of x_1 units of it, we will spend $5x_1$ on product one.

Similarly, we will spend $8x_2$ on product two. This will then lead to the budget constraint $5x_1 + 8x_2 \leq 60$.

Before we further discuss model formulations, it is mandatory that we familiarize ourselves with the basic assumptions of linear programming. In particular, there are three assumptions:

(1) deterministic property,
(2) divisibility, and
(3) proportionality (resulting in linearity).

First consider the deterministic property. *Deterministic* means that we assume that the structure of the problem as well as all parameters of the problem are assumed to be known with certainty. (The antonym of deterministic is *probabilistic* or *stochastic*). Clearly, it is not very realistic to assume that a model is deterministic when it, almost by definition and without very few exceptions, deals with future events and hence includes parameters that also relate to future events. However, while the original problem may be stochastic, our model could still be deterministic—just one of the many examples where a model is indeed a simplification of reality. We may get away with this simplification by using the trick of sensitivity analyses. As an example, if we can reasonably well estimate the future demand for a product to be between, say, 12 and 17, we may—at least for now—assume the demand to be known with certainty at the level of, say, 14 and solve the problem. Once that is accomplished, we employ sensitivity analysis to examine what happens to the solution if the demand were to decrease by one or two units. We would then continue to examine how the solution were to behave if the demand were to increase to 15, 16, or even 17. That way, we stay within the confines of deterministic models that are much easier to solve and still obtain information in case the demand is not at the level we first assumed.

The second assumption of linear programming deals with divisibility. It simply states that each variable, typically a quantity of some sort, can be expressed as any real number rather than an integer. Often, this is not satisfied. For instance, if we are in the business of making cans of beans, then the number of cans of beans, typically a variable in our planning model, will have to be an integer, as there is not much use in partial cans of beans that can obviously not be sold. However, it is very well known that dropping the assumption of divisibility creates major problems. In particular, if a linear program requires that variables can only assume integer values, we obtain an integer linear programming problem, which can be shown to be many times more difficult to solve as compared to a standard linear programming problem. Hence, the general rule is to assume divisibility as long as that can be justified. In the case of the cans of beans, we may assume divisibility and then simply round the solution, even though such procedure is well known to not necessarily result in an optimal solution. However, the potential loss of a part of a can of beans is so minuscule that it is not worth the effort to spend any time to find exact solutions. On the other hand, if we are planning the sale of houses, then

it may very well make a significant difference if we construct one house more or one house less. In summary, if the products we are dealing with are given in small numbers and are very valuable, we may have to drop the assumption of divisibility and solve a more difficult integer problem, while for low-value products in large numbers such an effort will not be required.

Finally, linear programming requires that all functions, objective functions as well as left-hand sides of constraints, are linear. The formal definition of linearity has already been discussed in Definition A.13 of this volume. The question here is whether or not the assumption of linearity is realistic in any given instance. Again, that will depend on the practical situation at hand. If, for instance, there are no quantity discounts for a product, i.e., the cost of a product is proportional to the quantity that we purchase, then the assumption of linearity is justified. If there are no economies of scale, it is. But clearly, many functions in real life are not linear. However, often—albeit not always—linearity is a reasonable approximation of reality. If it is not, then there is an old adage that applies to all of modeling: if the model is not sufficiently close to reality to make sense, then don't use it.

1.3 The Modeling Process

This section will guide readers through some of the intricacies of the modeling process, i.e., the process that starts at the present situation and ends with a fully implemented optimized solution. The main steps in the modeling process are as follows:

Step 1: Problem recognition
As straightforward as this step may sound, it is essential in the process. It requires that somebody in the organization realizes that it is not "business as usual," "we've always done it like that," and "we've never done it like that," but that it may be possible to improve the present situation. In addition to realizing that there is always room for improvement, it is necessary for the individual responsible for the firm or department under consideration to not only understand the workings of the department, but also the potential for improvement by the appropriate techniques. In other words, even if the manager of the department does not know exactly how to improve or optimize the system relating to his department, it is mandatory for the manager to be able to gauge as to whether or not there is some potential for improvement and what the possibilities of implementing any kinds of improvements really are.

Step 2: Convince the administration to model
As much as any one individual or group is convinced that improvements in the workings of a department are necessary, it is mandatory to obtain not just departmental approval or sanction, but also active support by superiors as well as departmental employees. In both cases, it will be necessary to "sell" the idea of optimization and the change that will result from the implementation of the new

solution. Selling modeling to superiors will require some reasonably clear ideas about the resources required in the process, as this will have a direct bearing on the costs of the undertaking. On the other hand, without active support of the people in the department that will be directly affected by the change, results will be boycotted and nothing will be accomplished (other than a waste of resources and a lot of bad blood).

Step 3: Collect information: stakeholders, structure, data
Once the modeling process has been approved, the modeler can go to work. Among the first steps will be to determine who will be the major stakeholders in the process are and what their objectives are. These do not have to be well-defined quantitative measures (and more often than not, they will not be), but general statements of utility functions. It will be one of the modeler's many tasks to quantify these utilities, determine surrogate or proxy criteria and obtain some agreement among the major stakeholders regarding the overall objective. In case the main decision makers cannot agree on a single objective, the modeler will have to resort to multiobjective optimization. Some of the appropriate tools for this case are discussed in Chapter 9 of this volume.

Once the objective has been determined at least in its general form, we need to define the structure of the model. By this we refer to the determination of the scope of the model, i.e., all of the subdepartments and issues to be included or excluded in the model. For instance, when optimizing a transportation system, one of the questions would be in how far the related inventory system will have to be included. Clearly, while it is desirable to include as many departments as possible in the model so as to avoid obtaining suboptima that may be good for one department but poor for another, such comprehensive models will make the system more expensive to model and more difficult to solve. What is appropriate in the specific case will depend on the judgment of the modeler.

The last step in this process is the collection of the data. This frequently underestimated task will, according to practitioners, always take longer than expected. Often, necessary data are hard to come by, due to the protective nature of subordinate managers or other employees who would like to guard their local fiefdoms or avoid having to change their habits. As an example, suppose that a modeler wants to determine the throughput of products at a workstation. He will ask the employee who operates the workstation for the appropriate figure and the reply may very well include the employee's fear to be forced to work harder, so that he may provide a number that is believable, but lower than it is in reality. Similarly, department managers asked about the performance of their department in terms of output or other measures may provide exaggerated numbers in order to look good. It is the modeler's duty to double-check and separate the fluff from reality. Other problems related to the task of data collection may relate to the unavailability of the type of data needed. For instance, census data may not cover the same region that a school district does, and customer surveys may not indicate the reasons for customer purchases or why they did not purchase a product.

Step 4: Build the model
This step (and the next) are what this book is all about. In particular, this step includes the definition of the variables, formulation of the objective(s) and the constraints. Chapter 2 of this volume will provide a number of typical scenarios that, by themselves, may be overly simplistic in their structure and size, but are indicative of some of the models encountered in practice. While there are usually multiple ways to formulate a model and as many ways to approach a problem as there are modelers, it is usually a good idea to start the formulation with the definition of the variables, i.e., those quantitative measures which can be influenced by the decision maker(s). Typical examples for variables are the number of items of a product manufactured, the amount of money allocated to a certain activity, the quantity shipped to a destination, and so forth. Sometimes, there are also so-called "logical variables," i.e., variables that indicate whether or not an activity is carried out or not. Such variables can assume only a value of zero or one; one, if we do engage in an activity, and zero if we do not. Such variables are not quantitative in the narrow sense, and we have dealt with them at length in Eiselt and Sandblom (2000). Whenever formulating an objective or a constraint, it is always useful to express the meaning of the expression first in terms of a regular sentence, which is subsequently (literally) translated into a mathematical function.

Step 5: Solve the model
This step appears straightforward. It includes the choice of the appropriate software, the inputting of the model, and its solution. Many of the remaining chapters in this book deal with this issue.

Step 6: Model validation
This important step will entail the examination of the solution obtained in the previous step. Are there outright errors to the model? Does the solution make sense? Could such a solution actually be implemented regarding potential internal and external resistance from individuals or organizations affected by the solution? In case the modeler is satisfied that the model is actually usable in the situation under consideration, we may move on to the next step. Otherwise, it is back to Step 4, in which the model can be revised. Note that this may include the collection of additional data or changes in the structure as shown in Step 3. The loop that consists of Steps 3 (or 4) to 6 may have to (and usually will) be repeated many times. This is what some authors mean when they refer to modeling being an interactive process.

Step 7: Model implementation
This is the final step in the process. It involves the modeler presenting his findings to the decision makers and those who have the power to approve the use of the solution. In order to ensure that the solution is actually used in practice, it is important that the modeler properly presents his findings. On the operational level it may be sufficient to simply write up a report with the findings, state the anticipated benefits of the new solution coupled with some thoughts regarding the

implementation of the solution, i.e., the changeover from the existing to the optimized solution, and that is it. However, this is definitely not sufficient in case of decisions to be made on the tactical or even strategic level. Here, it is again required that the modeler "sell" his solution to the stakeholders. The importance of this job is highlighted by the findings that even among those studies that were commissioned by the decision makers, i.e., for which they did put money and other resources up front, only a small fraction was ever implemented. One of the ways to make the selling of these findings more palatable is to present the "optimal" solution as one of the many possibilities to improve the situation, coupled with a thorough discussion of the assumptions that led to the solution, the changes that will result from changes in the data, and the robustness of the solution. After all, it is mandatory that the modeler do not take the position of the decision maker (if he is not the decision maker himself).

Having outlined some of the major steps of the process typically used in operations research, it becomes apparent that while there are many different foci in operations research (applications, theory, algorithms, and structures of models), it is modeling that sets operations research apart from other disciplines. As we have seen in the 7-step procedure above, modeling in this context refers to the translation of a real problem (referred to as "messes" by Ackoff (1974), one of the pioneers of the profession) into a well structured mathematical formulation. For a number of entertaining and instructive cases, see also Ackoff (1978).

As shown above, modeling is an interactive process. Of particular interest here is the loop that includes Steps 4-6. In order to explain the process, we refer again to the diet problem introduced at the beginning of the previous section. Suppose that the problem has been formulated and solved, and that we are now in the process of validating the solution.

For the sake of argument, assume that the solution suggests that the "optimal" daily food intake include three stalks of celery, four bunches of broccoli, two pounds of yoghurt, and five hamburgers (the latter as they provide cheap bulk). Two pounds of yoghurt are probably more than most people are willing to eat in a day, so the planner will have to include upper bounds on the quantity of that particular foodstuff. Once the model has been revised, the new model will be solved again.

What happens in the new solution is fairly easy to imagine: as the quantity of one type of food is reduced, the quantities of other types of food will have to increase in order to guarantee that sufficient amounts of nutrients are included in the diet. The new solution may include more celery, more broccoli, and possibly even more hamburgers. Such a solution is still not acceptable, so new constraints will have to be included, e.g., on the number of hamburgers in the diet. The process will circle until a diet has been created that is acceptable to both, the planner's budget and palate. It is clear that each time that additional constraints are introduced, the resulting optimal diet will not be cheaper (but typically more

expensive) than the previous diet. The planner will have to weigh the tastiness of his diet as it evolves, against its increasing price. Also note that it is very much in the planner's interest to enlarge his decision space by including additional foods that were not included previously. This process is shown in more detail in Chapter 2 of this book.

In general, we would like to comment on the changes that have occurred over the last decades as far as modeling is concerned. Given the lack of ubiquitous and fast computing some decades ago, it was mandatory for the modeler to think through the model as far as possible, anticipate potential problems and glitches, and basically try to present a model that would hopefully be as close as possible to what was desired. Since most realistic models are not formulated correctly the first time around, regardless how much care is taken to avoid errors, a lengthy error-finding, error-correcting, and re-solving process would ensue. In contrast, computing power is readily available today, which allows the user to start formulating a small model at first, that can easily be solved and corrected, if necessary. Then the modeler will add features to the model, re-solving the model frequently so as to keep errors confined to the smaller portions that were recently added. This simplifies the error-finding and error-correction tasks faced by the modeler. This process continues until the desired level of model sophistication has been reached.

As far as terminology is concerned, by a "solution" we mean a set of instructions or, if you will, an "action plan." In the context of the above diet example, a solution will include instructions of how much of each of the foodstuffs is included in the diet. These are then clear instructions of the type "eat one can of clam chowder per day." Associated with each solution is a consequence. Such a consequence is the value that the objective function takes when the set of instructions is followed. Again in the context of the diet problem, the consequence of a solution, i.e., eating a specific set of foods, will bear a consequence, i.e., a certain cost.

1.4 The Three Phases in Optimization

There are three major issues as far as any mathematical programming problem are concerned. They are usually arranged into three phases. They are *feasibility*, *optimality*, and *sensitivity*. In simple words, feasibility deals with the question whether or not the requirements, i.e., the constraints can be satisfied. If not, the modeler has to go back to the drawing board and rewrite the model, as no further processing is meaningful. Once a feasible solution has been found, we enter the second phase, which attempts to find an optimal solution. Since at least one feasible solution exists by assumption (otherwise we would not be in the second phase), there also exists at least one optimal solution. The second phase terminates with one such solution. Finally, the third phase examines what happens, if some of the parameters of the model change their values. Sensitivity analyses are also

referred to as *postoptimality analyses* and they can be recognized by their wording: they always include the terms "what – if."

Whenever a new model is formulated, it is usually a good idea to first determine roughly what the solution is going to be. In other words, we typically like to know what "ballpark" we are in. One way to do so is to perform a *break-even analysis* first. In terms of a profit-maximizing model, the break-even point is the point that separates a positive from a negative profit. Since profit P is defined as

$$Profit = Revenue - Costs,$$

where the revenue R is defined as unit price p times quantity q, and costs C are usually expressed as the sum of fixed costs C_f and variable costs C_v, and the latter of which are defined as unit costs c times quantity q, we can write

$$P = R - C = R - C_f - C_v = pq - cq - C_f = (p-c)\, q - C_f.$$

Assuming that the prices are parameters (i.e., fixed and known numbers) while the quantities are the variables, the break-even point can then formally be expressed as the quantity at which the profit equals zero or, equivalently, revenue equals cost. This results in the break-even quantity

$$q = \frac{C_f}{p - c}.$$

As an example, if it costs \$500 to set up the production and \$3 to make one unit (assuming there are no economies of scale), and units of the product that we make sells for \$5 (the demand is sufficiently high), then the break-even point is achieved where $q = 500/(5 - 3) = 250$ units. In other words, if we make and sell less than 250 units, we will lose money, while in case more than 250 units of the product are made and sold, we make a positive profit. This enables the decision maker to have a rough idea what his production figures have to be in order to make a profit.

In order to illustrate break-even analyses on a somewhat more elaborate example, consider the following

Example: The task at hand is to organize a scientific conference, which is assumed to be an annual event. In order to participate, conference registrants will have to pay a registration fee that is to be determined by the organizer. As customary, there will be different categories: regular attendees who register late (i.e., after a cutoff date or on-site), student and retired attendees who register late, regular attendees who register early, and student and retired attendees who register early. The charges for these four groups are to be determined and they are denoted by as $p_{r\ell}$, p_{re}, $p_{s\ell}$, and p_{se}, respectively. The number of attendees is, of course, also

unknown and the respective numbers are $x_{r\ell}$, x_{re}, $x_{s\ell}$, and x_{se}, respectively. The number of attendees replace the quantities in our general discussion above.

At past conferences of this type, the registration fees for the four categories were \$350, \$80, \$250, and \$50, respectively. The average attendance throughout the last few years has also been observed (we are taking an average so as to eliminate annual fluctuations, e.g., due to particularly attractive or unattractive conference venues). As we cannot assume that past attendance is a good guide to attendance figures at the planned conference, we use only the ratios of the past attendances between the four groups. Suppose that in the past, there were seven times as many regular attendees who registered late in relation to student and retired attendees who registered late, i.e., $x_{s\ell} = x_{r\ell}/7$. Similarly, we have observed that $x_{re} = x_{r\ell}/2$, and $x_{se} = x_{r\ell}/6$. Given that we want to keep the registration fees between the groups also at the same ratio, we have $p_{s\ell}/p_{r\ell} = 80/350$, $p_{re}/p_{r\ell} = 250/350$, and $p_{se}/p_{r\ell} = 50/350$.

As far as costs are concerned, we have to pay \$15,000 for the rental of the rooms, audio-visual equipment, and entertainment at the banquet. All of these costs will be incurred regardless of the number of conference participants. In addition, we will need \$25 for the conference kit (e.g., bag or binder, tag, CD with Proceedings, etc.) and \$60 for the reception, luncheon, and banquet.

At this point, we have eight unknowns (with the registration fees as variables and the attendance figures as unknown parameters) and a total of seven equations, *viz*., six ratios as outlined above and the break-even equation that requires that the revenue equals the cost, i.e.,

$$p_{r\ell}x_{r\ell} + p_{re}x_{re} + p_{s\ell}x_{s\ell} + p_{se}x_{se} = 15{,}000 + 85(x_{r\ell} + x_{re} + x_{s\ell} + x_{se}).$$

Expressing all fees and attendance figures in terms of $p_{r\ell}$ and $x_{r\ell}$ and solving for $x_{r\ell}$ results in the equation

$$x_{r\ell} = \frac{15{,}000}{1.4136\,p_{r\ell} - 153.8095}.$$

This makes it possible to construct a two-dimensional graph shown in Figure 1.1, in which we plot $p_{r\ell}$ against $x_{r\ell}$.

For instance, we can see that in order to break even, charging \$200 will require the attendance of 116.36 regular participants who register late (and, calculating the attendance figures of the other groups by using the above relations, requires a total of 211 participants). Such a requirement may be too optimistic, so that we examine the total number of required participants for a registration fee of \$250 for regular attendees who pay late. Similar calculations reveal that the total number of attendees now drops to 136, which may be more realistic. Similar figures for \$300

are 100 participants, for \$400 there must be at least 66 participants, etc. If conservative estimates indicate that about 80 participants can be expected, we could therefore decide to charge between \$350 and \$400 with a reasonable expectation to (at least) break even.

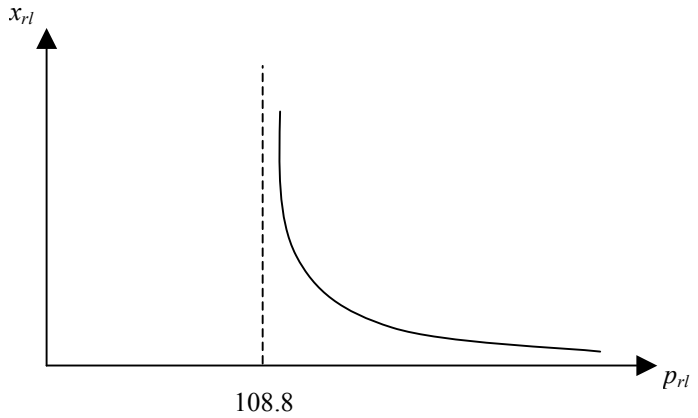


Figure 1.1

1.5 Solving the Model and Interpreting the Printout

This section will provide a simple scenario that is first formulated, and subsequently solved. We then provide a typical computer printout and we will demonstrate what information can be gleaned from it.

As a simple illustration of a formulation, consider the following

Example: Assume that in the context of a production problem there are two products, P_1 and P_2, that are manufactured on three machines, M_1, M_2, and M_3. Each product has to be processed on each of the three machines, but the order in which the products are processed on the machines is assumed to be immaterial. The two products sell for the (fixed) unit price of \$12 and \$27, respectively, while the capacities of the three machines are 15, 14, and 12 hours, respectively. Note that the capacities are expressed for the planning period, e.g., one day. It is important that all other capacity uses are then also expressed in the same units for the same planning period. The different capacities may result from different expected maintenance requirements. Assume now that it takes 9 minutes to process one unit of unit of P_1 on M_1, 6 minutes to process one unit of P_1 on M_2, and 10 minutes to process one unit of P_1 on M_3. The corresponding data for P_2 are 12, 14, and 18. Finally, it is required that at least 70 product units are made and sold.

The first step in the formulation is the definition of the variables. Here, the decision maker can only determine the quantities of the two products that he will produce. This leads to the definition of x_j as the number of units of product P_j that will be produced and sold, $j = 1, 2$. Note that the prices are parametric in this case, meaning that we either assume the firm to be a price follower, or assume that the firm does not feel that it can, at least not temporarily, change its prices. It is very important to define the variables carefully. For instance, had we defined x_j only as the number of units of P_j that is produced (rather than produced and sold), a number of important assumptions would have escaped us. Among them is the assumption that we will be able to sell everything that we make in the same period, i.e., there is no inventory buildup of unsold units and our customers are willing to purchase any amount at the specified price.

Start now with the objective function. In order to simplify matters, we will employ the decomposition principle. Simply stated, this principle takes a given expression that is deemed difficult to handle and subdivides it into smaller, more easily manageable, components. A reasonable objective in this example is to maximize the revenue. This can be decomposed into its two components (revenue that derives from P_1) and (revenue that derives from P_2). Since we know that revenue equals price times quantity, we can express the revenue from P_1 as the unit price of P_1 (which is known to be 12) multiplied by the quantity of P_1 that is made (which was formally defined as x_1). A similar argument can be made for product P_2, so that the profit which derives from P_2 can be written as $27x_2$. Once all individual parts have been determined, we enter the composition phase, in which the parts are put together into a single expression. In our example, the overall revenue equals the sums of the revenues that derive from the manufacturing and sales of both products, so that we obtain the objective

$$\text{Max } z = 12x_1 + 27x_2.$$

The constraints can be formulated in a similar fashion. First, we have to decide what type of constraints the model will have. In this example, it is easy to see that the first requirements deal with machine capacities, while the last demand constraint will require that a certain number of units is made. First consider the capacity constraints. Constraints of this type are always of the type "the capacity actually used cannot exceed the capacity that is available." Such a statement fits nicely into the mold of the constraint structure "left-hand side in relation to right-hand side" discussed above. The available capacities (formally, the right-hand side values) are the given parameters 15, 14, and 12 hours. The left-hand side values will express the capacities that are actually used in the production process. Consider one machine at a time and apply again the decomposition principle. On any machine, the capacity that is actually used is used for making products P_1 and P_2, there is no other use for the machine capacity. Considering M_1, we obtain the used capacity on that machine as the "capacity of M_1 used for making P_1 plus the capacity of M_1 used for making P_2." Since it takes 9 minutes to process one unit of P_1 on M_1, the time on M_1 that is used for the processing of P_1 is $9x_1$. Similarly, the

time on M_1 that is used for the processing of P_2 is $12x_2$, so that the capacity constraint of the first machine can be written as

$$9x_1 + 12x_2 \leq (15)(60),$$

where the right-hand side capacity must be expressed in the same units as the left-hand side (here: minutes). The other two capacity constraints are dealt with similarly.

It is very important to realize what the capacities actually mean. A capacity of, say, twenty-four hours means that each time a product is processed on the machine, a clock is turned on which runs until the processing is completed, at which time the clock is turned off. Once the total processing time has reached twenty-four hours, the capacity on this machine is exhausted and no further processing is possible within the planning period. While it may be tempting to interpret capacities as "hours of the day that a machine is available for processing," this is not true. Such interpretation would require the solution of an embedded scheduling problem, which is typically not done.

Finally, consider the demand constraint according to which at least 70 units of both products combined should be made. As the total number of products made and sold is $x_1 + x_2$, the requirement can be written as

$$x_1 + x_2 \geq 70.$$

The formulation of the entire problem is then

$$\text{P: Max } z = 12x_1 + 27x_2$$

$$\begin{array}{lrcll} \text{s.t.} & 9x_1 & + 12x_2 \leq (15)(60) & & \text{(Machine 1)} \\ & 6x_1 & + 14x_2 \leq (14)(60) & & \text{(Machine 2)} \\ & 10x_1 & + 18x_2 \leq (12)(60) & & \text{(Machine 3)} \\ & x_1 & + x_2 \geq 70 & & \text{(Demand constraint)} \\ & x_1, & x_2 \geq 0 & & \text{(Nonnegativity constraints)} \end{array}$$

Note that we have added so-called *nonnegativity constraints* to the formulation. As their name suggests, these constraints restrict the variables in the formulation to be nonnegative numbers. In most applications, such constraints are meaningful, so that they are typically added automatically by many software programs. Typical examples are variables that denote units to be manufactured, raw materials to be blended, inventory levels, and many others. All of these variables are naturally required to be nonnegative. On the other hand, variables that measure the throughput of an inventory may be positive (in case the inflow is larger than the outflow) or negative (if the inflow is less than the outflow). When constraints have been formulated, it is useful to briefly comment on the meaning of the constraints. This enables the user to know immediately what the constraints do

rather than having to go through the entire logic again, when it becomes necessary at a later stage to reconsider part of the formulation.

Before attempting to solve any given formulation, the modeler (in case the problem is solved manually) or the software may have to transform the formulation into a representation in *normal form*, i.e., in a form in which all constraints are written as equations. This requirement will depends on the type of software that is actually used. In order to write a problem in normal form, we first transform all constraints, so that their right-hand side values are nonnegative. We then add or subtract additional variables to and from the left-hand sides of the given constraints. More specifically, let the constraint be given again in its general form *LHS R RHS*. The additional variables are then *slack variables S*, *excess (surplus) variables E*, and *artificial variables A*, which are added to/subtracted from the left hand sides of the constraints according to Table 1.1.

Table 1.1

Type of constraint (Relation R)	Additional variable(s)	Constraint
$\leq$	Slack variable S	$LHS + S = RHS$
$\geq$	Excess (or surplus) variable E and artificial variable A	$LHS - E + A = RHS$
$=$	Artificial variable A	$LHS + A = RHS$

The meanings of the new variables can best be asserted by considering a $\leq$ constraint that restricts the use of a resource (i.e., a capacity constraint), and a $\geq$ constraint that provides a lower bound on the production of some good. Here, we will only discuss slack and excess variables, as artificial variables exist only for technical reasons. Their use is discussed in Chapter 3.

In a capacity constraint, a slack variable measures the difference between the actual use of a resource (the left-hand side) and the availability of the resource (the right-hand side value). As such it expresses the underuse of the capacity or, more precisely, the number of units of the resource that are left unused. Resources with zero slacks in the optimal solution identify bottlenecks in the schedule. On the other hand, given a constraint that expresses a production requirement, excess variables measure by how many units the actual production (the left-hand side) surpasses the requirement on the right-hand side.

In addition, we can assign values to each of the variables that provide a pricing system associated with all of the variables. To be more specific, we will consider a specific scenario similar to the production planning example above, in which we have a profit-maximizing objective coupled with resource constraints and demand constraints. The decision variables x_j denote the activity levels, i.e., the numbers of units of the products manufactured and sold, the slack variables S_i indicate the quantity of unused resources, and the excess variables E_k specify the quantity of

excess sales, i.e., the number of units that are delivered to the customers in excess of their demand. The costs or prices we associate with these variables will be referred to as c_x, c_S, and c_E.

The *reduced cost* or *opportunity cost* c_x can then be interpreted as the amount of money by which the price of the product will have to increase in order to make it profitable (and ensure its inclusion in the optimal production plan). Similarly, in a cost-minimization problem the dual prices will indicate the dollar amount by which the price of the product would have to decrease, so that it can be included in the optimal solution. Hence, in general the shadow prices state by how much money the given price will have to change, so that the activity in question will be included in the optimal solution. Given this definition, we can assert that the product of the optimal value of a decision variable and its opportunity cost will be zero. If the decision variable assumes a value of zero, the condition is naturally satisfied. If the value of the decision variable is positive, it is already included in the solution and no further price increase or cost decrease is required.

The *shadow prices* or *dual prices* c_S specify the value of one additional unit of the resource, i.e., the profit increase if one additional unit of the resource were available. Similarly, they indicate the profit decrease if one less resource unit were available.

Furthermore, the shadow prices or dual prices c_E indicate the value of one additional unit of production requirement, i.e., the profit decrease if the production requirement were to increase by one unit. Again, they also show the profit increase if the production requirement were to drop by one unit.

Similar to opportunity costs, the product of the value of a slack or excess variable and its associated shadow price must be zero. If a slack variable has a positive value, then the planner will not be prepared to pay additional funds to acquire more resources that are already available in (more than) sufficient quantities. Similarly, if an excess variable is positive, i.e., the present production level already exceeds the required level, then a modification of that level will not result in a change of profit.

Some authors, when referring to any of c_x, c_S, c_E, c_A call them *indicators* or *simplex multipliers*. Both terms are intimately related to the simplex method and its tableaus. This will be discussed in detail in Chapters 3 and 4.

We can now interpret the results of standard printout of the solution of a linear programming problem. For that purpose, we will use again as an example the production problem formulated above. In order to not tie our discussion to a specific solver, we show below a simulated printout that is similar to those found in many of today's packages. To facilitate a proper interpretation of the shadow prices, the printout specifies the type of constraint the individual shadow prices are attached to. Here, the abbreviation LE refers to a "≤" constraint, GE indicates

a "≥" constraint, and EQ denotes an equation. (Some printouts will use "<" to mean "≤" and ">" to refer to "≥".)

As an illustration, we use again the example discussed and formulated earlier in this chapter. For convenience, we restate the problem here.

P: Max $z = 12x_1 + 27x_2$

s.t.	$9x_1 + 12x_2 \le (15)(60)$	(Machine 1)
	$6x_1 + 14x_2 \le (14)(60)$	(Machine 2)
	$10x_1 + 18x_2 \le (12)(60)$	(Machine 3)
	$x_1 + x_2 \ge 70$	(Demand constraint)
	$x_1, x_2 \ge 0$	(Nonnegativity constraints)

A simulated printout will have the information shown in the box below.

VALUE OF THE OBJECTIVE FUNCTION: 877.50

DECISION VARIABLES	OPTIMAL VALUE	OPPORTUNITY COST
X1	67.5000	0.0000
X2	2.5000	0.0000

SLACK/EXCESS VARIABLES	CONSTRAINT TYPE	OPTIMAL VALUE	SHADOW PRICE
ROW 1	LE	262.5000	0.0000
ROW 2	LE	400.0000	0.0000
ROW 3	LE	0.0000	1.8750
ROW 4	GE	0.0000	6.7500

First the obvious. The values of the decision variables at optimum are 67.5 and 2.5, respectively, meaning that we should make and sell 67.5 units of our first product and 2.5 units of our second product. As both products have positive values in the optimal solution, i.e., they are manufactured in positive amounts, their opportunity costs are zero. This is hardly surprising as the opportunity costs measure by how much the prices of the products would have to be increased for the products to be manufactured. The value of the objective function specifies a profit of $877.50 for the production problem.

Equally important are the values of the slack and excess variables. Recall that the first three constraints were limiting the use of the three machines, meaning they are resource constraints. Hence the solver added slack variables to the left-hand

sides of these constraints, and the values of these slack variables indicate the number of unused machine hours. Here, we have a spare capacity of 262.5 hours for machine M_1 and 400 hours of unused time on machine M_2. There are, however, no hours left unused on machine M_3, indicating that M_3 constitutes a bottleneck in the production process. The identification of bottlenecks in the production process is crucial, as it will lead to the acquisition of additional capacities or the sale of existing capacities in the future. Whenever a constraint has a zero slack or excess variable that has a value of zero at optimum, we say that this constraint is *binding* at optimum. Finally, the demand constraint in row 4 is also binding, indicating that the customer demand is satisfied, but not additional units are delivered.

Consider now the shadow prices. The shadow prices associated with the first two constraints (i.e., machines) are zero. This indicates that the profit would increase by zero if additional units of the first two resources, i.e., additional machine hours, were available. This is reasonable considering the fact that there exist unused capacities on these machines at optimum. On the other hand, the shadow price associated with row 3, the row that corresponds to machine M_3, is \$1.875, showing that each additional hour of time on M_3 (within reason, a concept further elaborated upon in Chapter 6), will increase the profit by \$1.875. On the other hand, the shadow price of constraint 4 (the demand constraint) indicates that increasing (decreasing) the lowest acceptable demand from its present value of 70 will reduce (increase) the profit by \$6.75. In other words, if our customers could agree to reduce their demand for at least 70 units to 69, we would deliver a different mix of products (more specifically, 65.25 units of product P_1 and 3.75 units of product P_2) and save \$6.75 in the process. On the other hand, if the customer were to demand a total of 71 units instead of 70, we should charge \$6.75 *in addition to* the price of the products.

2 APPLICATIONS

This chapter will provide a number of classes of applications of linear programming problems. We should note that these are mere sample problems that are designed to give readers a flavor of the application, rather than a model that includes all of the bells and whistles usually found in real applications. We have already discussed an example of the first large class of applications in the Introduction, *viz*., production problems. Readers may recall that the simple model formulated in the introduction has its decision variables defined in terms of the number of quantity units of a product that are made and sold. Much more frequently, such variables will have to be decomposed into variables that measure the number of units that are made, while other variables express the number of units that are sold, with the unsold units being left in inventory. A sizeable production – inventory problem is presented as a mini case study in Section 10 of this Chapter.

2.1 The Diet Problem

The first known instance of the diet problem actually predated the formal development of linear programming. A solution to a small problem was first calculated without linear programming by (later Nobel laureate) George Stigler (1945), whose guess was shown to be very close to optimal. An interesting account is provided by Garner Garille and Gass (1981). The basic idea behind the problem is simple: choose quantities of foods so as to satisfy nutritional requirements and ensure that the price of the resulting diet is within reason. Two versions of the problem can, at least in theory, be thought of: either we minimize the cost of the diet and ensure by way of the constraints that some nutritional constraints are satisfied, or we maximize the nutritional content of the diet subject to a budget constraint. While the latter approach appears appealing, it is impractical, as it requires the user to find an expression for the nutritional content, which is a blend of many different nutrients whose quantities are measured in different units and cannot be added.

Formally, the cost-minimization version of the problem can be defined as follows. Define x_j as the quantity (e.g., the number of servings, ounces, pounds, or any other measure) of the j-th food in the diet, let a_{ij} denote the amount of nutrients of type i in one unit of food j, define c_j as the cost for one serving of food j, and denote by $\overline{b}_i$ ($\underline{b}_i$) the largest (smallest) acceptable quantity of nutrients of type i in the diet. The problem can then be formulated as

$$\text{P: Min } z = \sum_j c_j x_j$$

$$\text{s.t. } \underline{b}_i \le \sum_j a_{ij} x_j \le \overline{b}_i \quad \forall \ i$$

$$x_j \ge 0 \ \forall \ j.$$

Some of the nutrients may not have an upper or a lower bound, in which case we set the appropriate parameters $\overline{b}_i$ and $\underline{b}_i$ to a sufficiently large and a zero value, respectively. Also note the necessity of the nonnegativity constraints in this formulation as we can ingest only positive quantities of food.

In the remainder of this section, we will first formulate a small version of the problem with some real data and then demonstrate the interactive modeling process. This may serve as an example for modeling in general, as it is by no means restricted to the diet problem. Table 2.1 shows the nutritional contents of six foods, three of which were taken from different fast food chains, two are beverages, and the remaining food are bagels. We also consider nine nutrients that are among the standard nutritional components typically considered in diets. The nutritional information has been obtained from the appropriate websites and Walford (2007) for the nutritional content of foods.

The prices have been obtained directly at fast food outlets and a supermarket. The cost-minimizing model was then solved. Its solution included 0 hamburgers, 0 banana splits, 2.66 cups of broccoli, 4.28 bagels, 0.98 cups of milk, and 9.16 cups of orange juice for a total cost of $5.93.

The meal plan is certainly cheap. Notice that the protein, fiber, the lower bound on the calorie constraints, and the vitamin A constraint are binding at optimum, meaning that the present solution includes just about enough protein, fiber, calories, and vitamin A.

The plan does not look bad, but includes excessive amounts of orange juice, so that we may now try to limit ourselves to no more than four cups of orange juice. Hence we add the constraint $x_6 \le 4$ to the above model and resolve the problem.

Table 2.1

Food	Price	Protein	Fiber	Carbs	Calories	Chol-esterol	Vitamin A	Vitamin C	Saturated fat	Sodium
Big Leo Burger	\$3.29	24	3	44	530	65	10%	4%	10	1,020
Banana Split	\$3.99	8	3	96	510	30	3%	33%	8	180
Raw broccoli, 1 cup	45¢	3	3	5	25	0	27%	137%	0	24
Whole grain bagel	48¢	5	3	27	140	0	0	0	0	270
2% Milk, 1 cup	40¢	8.5	0	13	130	20	10%	4%	3	125
Orange juice, 1 cup	25¢	2	1	27	110	0	2%	100%	0	0
RDA		≥ 56	≥ 30	≥ 130	∈ 1.8K, 2.2K	≤ 300	≥ 100%	≥ 100%	≤ 24	≤ 2,400

RDA: recommended daily allowance

The solution now includes 0 hamburgers, 0 banana splits, 2.25 cups of broccoli, 6.41 bagels, 3.12 cups of milk, and 4 cups of orange juice for a total cost of $6.34.

The modified requirements that led to this new solution have increased the price somewhat, but not much. In this solution, the fiber constraint, the lower bound on the calories, and the vitamin A constraint are binding. While the quantity of orange juice is now reasonable, there are now too many bagels in the solution. We now add the constraint that the solution should not include more than four bagels, i.e., $x_4 \leq 4$.

The new optimal solution then includes 0 hamburgers, 0 banana splits, 4.67 cups of broccoli, 4 bagels, 5.26 cups of milk, and 4 cups of orange juice for a total cost of $7.12.

Again, we notice a slight increase in price, but the diet is still cheap. The fiber constraint is still binding and so is the lower bound on the calories. Also, we may feel that nine cups of liquid are excessive, so that we add a constraint that limits our intake of liquids to no more than six cups per day. This constraint is formulated as $x_5 + x_6 \leq 6$.

The new optimal solution then includes 0.84 hamburgers, 0 banana splits, 3.83 cups of broccoli, 4 bagels, 2 cups of milk, and 4 cups of orange juice for a total price of $8.20.

The fiber constraint is still binding, as is the lower bound on the calories. At this point we may feel quite satisfied with the diet, but may wish to limit the quantity of broccoli in the diet. In order to do so, we add a constraint that limits our intake of broccoli to two cups, i.e., we add the constraint $x_3 \leq 2$ to the model . Solving the revised problem does, however, not result in a feasible solution (the lack of fiber content in the other foods is the major problem). In order to restore feasibility, we relax the new constraint and only require that the diet not include more than three, rather than two, cups of broccoli, i.e., we replace the constraint $x_3 \leq 2$ by $x_3 \leq 3$.

The new optimal solution then includes 1.12 hamburgers, 0.53 banana splits, 3 cups of broccoli, 4 bagels, 0 cups of milk, and 4 cups of orange juice for a total cost of $10.13. In this solution, fiber and sodium are now at their allowable bounds. This diet appears to be reasonable: one hamburger, one banana split every other day, three cups of broccoli, and four glasses of orange juice. The price of $10.13 also appears to be reasonable (even though adding the restriction on broccoli increased the price by 23½%). We also notice in the process that the fiber constraint has been tight all along, suggesting that we add some other, fiber-rich, food items to our selection.

The discussion above was meant to not only introduce the diet problem, one of the earliest and still important applications of linear programming, but also provide some insight into the modeling process.

2.2 Allocation Problems

Allocation models belong to the class of linear programming problems that includes arguably the most practical applications. All allocation models have in common that they attempt to allocate a scarce resource so as to optimize the consequence of that allocation. More specifically, we may have to allocate money to different types of investments in order to establish a retirement fund, (machine) time to individual goods in the manufacturing of products, manpower to sales and administrative positions, fuel, or other resources to different economic activities, and so forth. These allocations are to be performed so as to maximize their profit, minimize their cost, or optimize other efficiency criteria specified by the decision maker.

As an illustration, consider the following

Example 1 (an agricultural allocation model): A farmer owns 1,000 acres of more or less homogeneous farmland. His options are to breed cattle, or plant wheat, corn, or tomatoes. It takes four acres to support one head of cattle. Annually, 12,000 hours of labor are available. (For simplicity, we will assume here that these 12,000 hours could be used at any time during the year, i.e., through hiring casual labor during seasons of high need, e.g., for harvesting). Table 2.2 provides information regarding the profit, yield, and labor needs for the four economic activities.

Table 2.2

	cattle	wheat	corn	tomatoes
Profit	\$1,600 /head	\$5/bushel	\$6/bushel	50¢/lb
Yield per acre	¼ heads/acre	50 bushels	80 bushels	1,000 lbs
Annual labor requirement	40 hrs/head	10 hrs/acre	12 hrs/acre	25 hrs/acre

Furthermore, it is required that at least 20% of the farmland that is cultivated in the process must be used for the purpose of cattle breeding, at most 30% of the available farmland can be used for growing tomatoes, and the ratio between the amount of farmland assigned to growing wheat and that left uncultivated should not exceed 2 to 1.

As usual, the first step in formulating the problem is to define variables. In allocation problems, we typically formulate variables x_j as the quantity of the scarce resource allocated to the j-th activity. In case of multiple scarce resources, there are different ways to formulate the problem. In this example, we have two scarce resources, *viz*., land and labor. We choose land for our formulation, so that the variables will be defined as the amount of farmland that is devoted to the j-th activity. The model is then

$$\text{P: Max } z = 1{,}600(¼)x_1 + 5(50)x_2 + 6(80)x_3 + ½(1{,}000)x_4$$

whose units are

$$[\$] = [\$/\text{head}][\text{heads/acre}][\text{acres}] + [\$/\text{bushel}][\text{bushel/acre}][\text{acres}] + [\$/\text{bushel}][\text{bushel/acre}][\text{acres}] + [\$/\text{lbs}][\text{lbs/acre}][\text{acres}]$$

$$\begin{aligned}
&\text{s.t.} \quad x_1 + x_2 + x_3 + x_4 \le 1{,}000 && (1)\\
&40(¼)x_1 + 10x_2 + 12x_3 + 25\,x_4 \le 12{,}000 && (2)\\
&x_1 \ge 0.2(x_1 + x_2 + x_3 + x_4) && (3)\\
&x_4 \le 0.3(1{,}000) && (4)\\
&x_2/(1{,}000 - x_1 - x_2 - x_3 - x_4) \le 2/1 && (5)\\
&x_1, x_2, x_3, x_4 \ge 0.
\end{aligned}$$

Constraint (1) ensures that we do not use more farmland than is actually available, constraint (2) restricts the use of labor, constraint (3) guarantees that at least 20% of the land that is used is actually used for cattle breeding, constraint (4) restricts the land for tomato planting to no more than 30% of the available land, and constraint (5) restricts the required 2 to 1 ratio.

Notice that we have to pay close attention to the different units in which the items in this formulation are expressed, which is why we have shown the units of the parameters and variables used in the objective function. Also note that the ratio constraint is actually nonlinear in its current formulation. However, a few routine transformations result in a linear constraint. Cleaning up the model results in

$$\text{P: Max } z = 400x_1 + 250x_2 + 480x_3 + 500x_4$$

$$\begin{aligned}
\text{s.t. } & x_1 + x_2 + x_3 + x_4 \le 1{,}000\\
& 10x_1 + 10x_2 + 12x_3 + 25x_4 \le 12{,}000\\
& .8x_1 - .2x_2 - .2x_3 - .2x_4 \ge 0\\
& x_4 \le 300\\
& 2x_1 + 3x_2 + 2x_3 + 2x_4 \le 2{,}000\\
& x_1, x_2, x_3, x_4 \ge 0.
\end{aligned}$$

The optimal solution is $\bar{x}_1 = 200$, $\bar{x}_2 = 0$, $\bar{x}_3 = 769.2308$, $\bar{x}_4 = 30.7692$ acres with an optimal profit of $\bar{z} = 464{,}615.4$.

It is also worth mentioning that the variable x_1 would in reality be constrained to be integer multiples of 4, as the number cows must be integer and each cow requires four acres. For simplicity, we have deleted such considerations here. If this were a concern, then we would redefine x_1 as the number of heads of cattle. In the above formulation, the only change is that x_1 is replaced by $4x_1$ (with the specification that x_1 be integer).

Another possibility is to define the variables in terms of labor, the other scarce resource. In particular, we can define x_j as the hours of labor devoted to activity j. The (cleaned up) formulation is then

$$\text{P: Max } z = 40x_1 + 25x_2 + 40x_3 + 20x_4$$

$$\begin{aligned}
\text{s.t. } & .1x_1 + .1x_2 + .0833x_3 + .04x_4 \le 1{,}000 \\
& x_1 + x_2 + x_3 + x_4 \le 12{,}000 \\
& .08x_1 - .02x_2 - .0167x_3 - .008x_4 \ge 0 \\
& .04x_4 \le 300 \\
& .2x_1 + .3x_2 + .1667x_3 + .08x_4 \le 2{,}000 \\
& x_1, \quad x_2, \quad x_3, \quad x_4 \ge 0.
\end{aligned}$$

It is apparent that the two formulations are simple transformations from each other. In particular, there are ratios of 10:1, 10: 1, 12:1, and 25:1 between the variables in the two formulations. This results from the conversion from [hours/acre], which are 10, 10, 12, and 25. The solution does, of course, reflect this as well with $\bar{x}_1 = 2{,}000$, $\bar{x}_2 = 0$, $\bar{x}_3 = 9{,}230.7656$, $\bar{x}_4 = 769.2344$ hours with an optimal profit of $\bar{z} = 464{,}615.3$, the same as in the previous formulation.

Due to their widespread occurrence, we will provide another allocation model from portfolio analysis.

Example 2 (a portfolio selection problem): An investor has $1 million to invest in any combination of bonds, stocks, term deposits, a savings account, real estate, and gold. The anticipated (or known) interest rates, the risk factors (where a high number indicates a high risk) and the expected increase in the value of the investment are shown in Table 2.3:

Table 2.3

Type of investment	Interest (in %, annually)	Risk factor	Expected annual increase in value
Bonds	5%	3	0%
Stocks	2%	10	7%
Term deposits	4%	2	0%
Savings account	3%	1	0%
Real estate	0	5	7%
Gold	0	20	11%

For instance, real estate does not yield any interest (the projects are not rental properties), but its value is expected to increase by 7% per year. On the other hand, stocks yield an average of 2% in interest, and in addition to that, they are expected to increase in value by 7% for a total of 9% p.a. The other numbers are to be interpreted similarly.

The objective is to maximize the amount that is expected to be available in a year's time subject to the following restrictions:

- Of the total amount of money invested, at least 30% must be invested in bonds, not more than 10% in stocks, and at least 10% in term deposits and/or savings accounts.
- Up to 50% of the total money invested in real estate may be borrowed against in the form of a mortgage at an interest rate of 6%. The amount borrowed cannot exceed $150,000.
- The average risk factor of the investment cannot exceed 4.5.
- The average annual interest should be at least 2.5%.
- The amount of money invested in gold cannot exceed $100,000 or 8% of the total amount of money available, whichever is smaller.

In this case, money is the only scarce resource. Defining x_j as the amount of money allocated to the j-th investment and x_7 as the amount borrowed, we can write

$$
\begin{aligned}
&\text{P: Max } z = 1.05x_1 + 1.09x_2 + 1.04x_3 + 1.03x_4 + 1.07x_5 + 1.11x_6 - 1.06x_7\\
&\text{s.t. } x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \le 1{,}000{,}000 + x_7\\
&x_1 \ge 0.3(x_1 + x_2 + x_3 + x_4 + x_5 + x_6)\\
&x_2 \le 0.1(x_1 + x_2 + x_3 + x_4 + x_5 + x_6)\\
&x_3 + x_4 \ge 0.1(x_1 + x_2 + x_3 + x_4 + x_5 + x_6)\\
&x_7 \le 0.5x_5\\
&x_7 \le 150{,}000\\
&\frac{3x_1 + 10x_2 + 2x_3 + 1x_4 + 5x_5 + 20x_6}{x_1 + x_2 + x_3 + x_4 + x_5 + x_6} \le 4.5\\
&\frac{5x_1 + 2x_2 + 4x_3 + 3x_4}{x_1 + x_2 + x_3 + x_4 + x_5 + x_6} \ge 2.5\\
&x_6 \le 100{,}000\\
&x_6 \le 0.08(1{,}000{,}000 + x_7)\\
&x_1, x_2, x_3, x_4, x_5, x_6, x_7 \ge 0.
\end{aligned}
$$

The optimal solution of this problem is:

Bonds	$437,000
Stocks	$115,000
Term deposits	$115,000
Savings account	$0
Real estate	$478,400
Gold	$4,600
Loan	$150,000,

so that the total return is \$1,061,794. If the interest for the loan were to increase by at least 0.1560 percentage points, the solution would change. (The information that leads to this number is discussed in Chapter 6). For instance, if the mortgage rate were to increase to 6.5%, we would no longer take out any loan, and the total return would be \$1,061,560, a decrease of only 0.022%. More specifically, the solution would be

Bonds	\$380,000	(−13%)
Stocks	\$100,000	(−13%)
Term deposits	\$100,000	(−13%)
Savings account	\$0	(±0%)
Real estate	\$416,000	(−13%)
Gold	\$4,000	(−13%)
Loan	\$0	(−100%),

where the numbers in parentheses indicate the changes of the new solution as compared to those of the previous solution. It is apparent that while the changes in the individual investments are significant, the return hardly changes at all.

2.3 Cutting Stock Problems

Cutting stock problems were first described in the early days of linear programming. Gilmore and Gomory (1961) were the first to formulate cutting stock problems (or, equivalently, stock cutting or trim loss problems). The problem in its simplest form can be described as follows. Given materials that are available in certain shapes and sizes, cut them in order to generate certain desired shapes and sizes, so as to minimize some objective such as cost. Depending on the type of material and cutting under consideration, we distinguish between one-, two-, and three-dimensional cutting problems. For instance, if we want to cut newsprint to different lengths, then—although the paper is certainly two-dimensional—this is a one-dimensional cutting problem, as the width of the paper remains the same, while only its length varies. Here, we first formulate a one-dimensional cutting stock problem and later generalize to two dimensions.

In order to simplify the terminology, assume that we are cutting rods to desired lengths. Being a one-dimensional problem, all rods are of the same diameter and only their respective lengths differ. On the one hand, there are different lengths of *available rods* that we have in stock, while on the other hand, there are lengths of *desired rods* that we or our customers demand. A *cutting plan*, consisting of *cutting patterns*, will determine how to generate the desired rods from the available rods. In order to formalize, suppose that there are m available lengths for which there are supplies of s_i, i=1, …, m. On the other hand, there are p desired lengths for which the demands are d_k, k=1, …, p. The cutting plan is assumed to consist of n distinct cutting patterns that are enumerated from j=1 to n.

Furthermore, a_{kj} denotes the number of desired lengths k that are generated by cutting according to pattern j, c_j denotes the cost of cutting pattern j, and the set T_i includes all patterns j that use the available length i. In addition to cutting available stock in order to generated the desired lengths, it may also be possible to purchase desired lengths directly. The cost per unit of length k is then c_k^*.

We can then define the required variables. The first set of variables determines how the cutting is going to take place. In particular, define x_j as the number of times we cut the j-th pattern. The variables y_k are defined as the number of desired lengths k that will be purchased in addition to those being cut. We can then formulate the cutting stock model as follows.

$$\text{P: Min } z = \sum_j c_j x_j + \sum_k c_k^* y_k$$

$$\text{s.t. } \sum_{j \in T_i} x_j \le s_i \quad \forall\ i = 1, \ldots, m$$

$$\sum_j a_{kj} x_j + y_k \ge d_k \quad \forall\ k = 1, \ldots, p$$

$$x_j, y_k \in \mathbb{N}_0 \quad \forall\ j = 1, \ldots, n;\ k = 1, \ldots, p.$$

In this formulation, the objective function minimizes the sum of cutting and purchasing costs. The first set of constraints guarantees that the number of available rods that are cut will not exceed the number of rods of that length that are actually in stock. The second set of constraints ensures that the number of desired rods that are cut from stock plus those that are purchased, will satisfy the demand.

In order to further explain the formulation, consider the following

Example: A firm has an inventory of one hundred and twenty 20-ft rods, one hundred and sixty 15ft rods, and forty 8-ft rods. Their customers demand two hundred 10-ft rods, and two hundred and fifty 6-ft rods. Each cut costs 30¢, while buying the rods costs 50¢ and 25¢, respectively, for the 10-ft and 6-ft rods. Given the above information, we can now determine the cutting plan shown in Figure 2.1.

This cutting plan includes only patterns that cut as many of the desired lengths as possible out of each of the given lengths. This appears intuitively appealing, as, for instance, it would seem meaningless to include a pattern that cuts one 20' rod to just a single 6' rod. Such patterns may, however, be meaningful as the 14' rod that is left over, may be a useful input in the future when different lengths are in demand, whereas the 2' length that results from pattern 3 will almost certainly be waste.

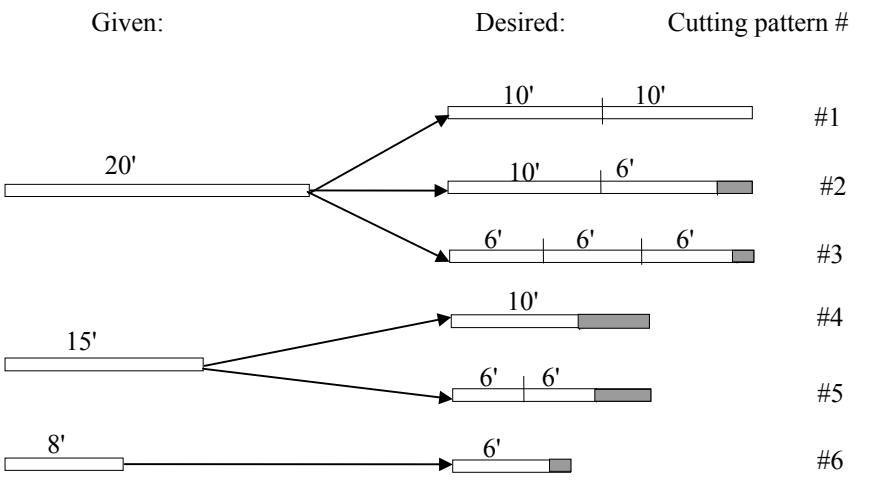


Figure 2.1

Define now x_j, $j = 1, \ldots, 6$ as the number of times a given rod is cut according to the j-th pattern. Furthermore, let y_1 and y_2 denote the number of 10 ft & 6 ft rods that are purchased in addition to those cut. As far as cutting costs are concerned, pattern 1 uses one cut and hence costs 30¢, pattern 2 requires two cuts and therefore costs 60¢, and so forth. The problem can then be formulated as follows.

$$\text{P: Min } z = 0.3x_1 + 0.6x_2 + 0.9x_3 + 0.3x_4 + 0.6x_5 + 0.3x_6 + .5y_1 + .25y_2$$

$$\begin{array}{lrrrrrrrrr}
\text{s.t.} & x_1 & + x_2 & + x_3 & & & & & & \leq 120 \\
 & & & & x_4 & + x_5 & & & & \leq 160 \\
 & & & & & & x_6 & & & \leq 40 \\
 & 2x_1 & + x_2 & & + x_4 & & & + y_1 & & \geq 200 \\
 & & x_2 & + 3x_3 & & + 2x_5 & + x_6 & & + y_2 & \geq 250 \\
 & x_1, & x_2, & x_3, & x_4, & x_5, & x_6, & y_1, & y_2 & \geq 0
\end{array}$$

The optimal solution to this problem suggests that we cut the first pattern hundred times (thus generating the required two hundred 10 ft rods) and purchase the remaining two hundred and fifty 6 ft rods for a total cost of $92.50. This leaves us with twenty 20 ft rods, hundred and sixty 15 ft rods, and forty 8 ft rods, all uncut and usable as possible inputs in the future. Notice that this solution results in no waste at all. The reason for the purchase of the 6 ft rods can easily be seen: patterns 3, 5, and 6 generate three, two, and one 6 ft rods, respectively, with three, two, and one cuts, while pattern 2 produces one 6 ft rod and one 10 ft rod. The prices for cutting the rods are then 30¢ per rod in each case (assuming that the cutting costs for pattern 2 are equally attributed to the 6 ft rod and the 10 ft rod), while a purchasing price of a 6 ft rod is only 25¢.

Ignore now the option of purchasing rods. In the model this means simply that the variables y_1 and y_2 are deleted, i.e., set equal to zero. Solving the remaining model results in cutting the first pattern 100 times, the third 20 times, the fifth 75 times, and the sixth 40 times. The total cost are now \$105, indicating that the option that allows us to purchase rods is worth \$12.50.

Consider now the same problem, where, again, we do not allow the option of purchasing additional rods, and furthermore, let the objective be the minimization of waste, which is defined as the simple leftovers. Notice that this poses the question of measuring waste. While this appears quite straightforward in that individual lengths of waste can simply be added up, this may not be a proper procedure. While a 1-ft piece that is left by the cutting process may indeed by waste, a 5-ft piece that cannot be cut into any length that is in demand right now, could potentially become useful input in the future and should therefore not be properly counted as five times the waste of a single 1-ft piece. In such a case, the counting of the waste would be adjusted accordingly by setting it to zero in the formulation, whenever it has been decided that the pieces that are not used in this period may be usable later. However, if the decision maker deems a pattern that generates large pieces that will remain unused in this planning period as unacceptable, then such patters will simply be ignored and deleted from the cutting plan.

In the numerical example below, we ignore this argument and determine total waste as the sum of all pieces that remain unused in the cutting process. Hence, the six patterns in our example generate 0, 4, 2, 5, 3, and 2 ft of waste, respectively. The problem without the option of purchasing rods and with the minimization of waste then simply deletes the variables y_1 and y_2, and the new objective function is

$$\text{Min } z = 4x_2 + 2x_3 + 5x_4 + 3x_5 + 2x_6.$$

The optimal solution of this problem is quite different to the original one. While we still cut the first pattern 100 times (generating two hundred 10 ft rods), we now cut the third pattern 20 times (thus generating sixty 6 ft rods), and the fifth pattern 95 times (thus generating hundred and ninety 6 ft rods), so that the required 200 and 250 rods result. The total waste is 325 ft which will cost \$105, the same amount as the solution that minimizes the costs.

The main difficulty related to the cutting stock problem is not the formulation by itself, but the setting up of the cutting plan. In most practical applications, the number of cutting patterns is astronomical. The reason for this are not only the number of different lengths that are either available or in demand, but also the relation between the lengths of available and demanded lengths. Consider, for instance, the same example as above, but assume that the longest available length were not 20 ft, but 200 ft instead. The number of cutting patterns would increase tremendously on account of this change.

Consider now two-dimensional cutting stock problems. Interestingly enough, the formulation does not change at all. The only difference is that it is now much more difficult to enumerate the possible cutting patterns. Even if we were to ignore arrangement of desired shapes that are crooked or somewhat irregular, the problem remains difficult. As an illustration, consider the following

Example: A firm has 10-ft by 10-ft sheets of metal that are to be cut into 3-ft by 5-ft and 2-ft by 6-ft pieces. Given only a single two-dimensional shape and two desired shapes (which, for simplicity, we will refer to as types T_1 and T_2), the problem would appear to be easy. A few simple patterns are shown in Figure2.2. All patterns shown here have the desired pieces arranged with their edges parallel to the edges of the available piece.

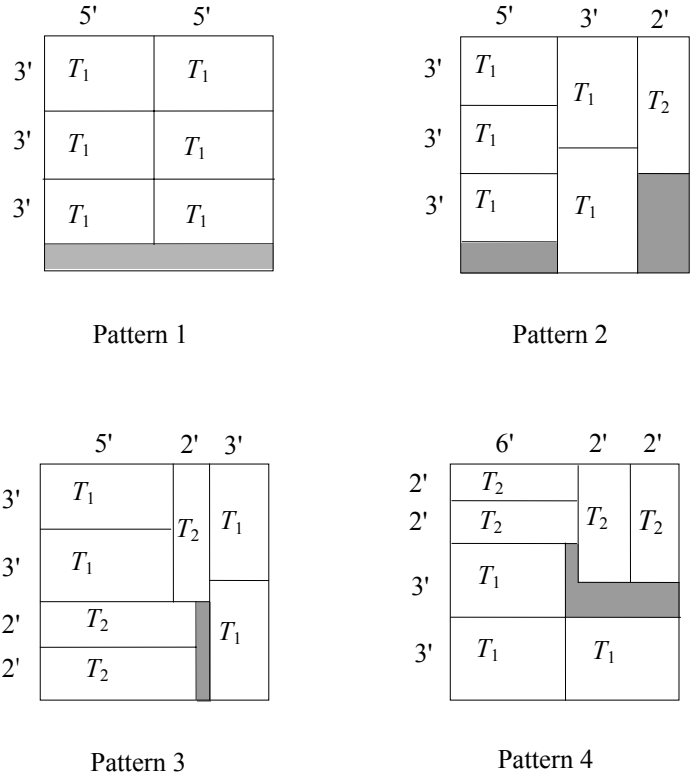


Figure 2.2

One possibility to evaluate the patterns and their respective efficiencies is the usage of the available material. The available material is 100 sq. ft., while the four patterns use 90, 87, 96, and 93 sq. ft., respectively. So it would appear that the third pattern were the most efficient. While this may be true theoretically, that

pattern is very difficult to cut. For that purpose, we would like to introduce the concept of *guillotine cuts*. These cuts have the property that they cut a given piece right through without stopping somewhere in the middle. Most practical applications use guillotine cuts exclusively. So, the first question is whether or not a given pattern can actually be cut by using guillotine cuts. While this does not appear to be the case of pattern 3, it is, albeit in a tedious way. At first, we would cut the original piece into a 10' × 7' and a 10' × 3' piece. The latter piece can then easily be cut into two T_1 pieces and that part is done. The 10' × 7' piece can then be cut into 4' × 7' and 6' × 7' pieces, which then can easily be cut further into the desired pieces. As a result, we have determined that Pattern 3 can indeed be generated by guillotine cuts. Note that Pattern 4 can also be cut by guillotine cuts. However, the cutting width has to be adjusted often with the strong possibility of operator error. Therefore, simple patterns such as Patterns 1 and 2 are strongly preferred, even if their theoretical efficiency, as measured by their usage of material, make them appear inferior.

2.4 Employee Scheduling

This section presents an admittedly simplistic scheduling problem that assigns employees to shifts. Typical examples would be bus drivers, hospital nurses, or other shift workers. It must be noted that the formulation presented below does not distinguish between individuals: it only states that, for instance, five drivers are assigned to a specific shift, but it does not indicate who these drivers are and if their allocation would cause a violation of rules set out in their collective agreement (e.g., working shifts too close together, working to many hours, or similar regulations). It also does not take into consideration the possibility of different abilities of the employees. This problem can be seen as the first stage in an employee scheduling systems. The second phase would then be the allocation of individual employees to shifts. This can, of course, also be done in a single stage as shown below.

To formalize, assume that the relevant time slots have been numbered $j = 1, 2, \ldots, n$, and that the smallest number of employees that have to be present during time slot j is denoted by b_j. For notational convenience, we define $b_{n+1} := b_1$. We can then define variables x_j that denote the number of employees that start working in time slot j. Again, for convenience we define $x_{n+1} = x_1$. It now depends how the time slots are defined in relation to the length of the shifts that employees have to work. For instance, if the time slots are defined as four hours (as the need for employees is constant during any 4-hour period) and each shift is 8 hours, then we note that an employee who starts work at the beginning of time slot j will then be available during time slots j and j+1. If the time slots are defined on a two-hour basis, then an employee who starts to work in time slot j will work during time slots j, j+1, j+2, and j+3. Assuming that time slots are four hours and shifts last eight hours and given the objective to minimize the number of employees that will be needed, we can formulate the problem as

$$\text{P: Min } z = \sum_j x_j$$
$$\text{s.t. } x_j + x_{j+1} \geq b_{j+1} \ \forall\, j{=}1,\ldots, n$$
$$x_j \geq 0 \ \forall\, j.$$

Notice the special structure of the matrix of coefficients. It is a so-called Hoffman-Kruskal matrix (see Hoffman and Kruskal, 1956)

$$\mathbf{A} = (a_{ij}) \text{ with } a_{ij} = \begin{cases} 1, \text{ if } i = j \\ 1, \text{ if } i = j+1 \\ 1, \text{ if } i = 1 \text{ and } j = n \\ 0 \text{ otherwise} \end{cases}.$$

It is easy to demonstrate (e.g., by way of evaluating the determinant with respect to the first row) that **A** has a determinant of two, if n is odd and a determinant of zero, if n is even. With four-hour time slots, a day has six time slots, so that n is even if we plan for one or multiple full days. This ensures integrality for all integer requirements b_j without us specifically requiring it.

In order to illustrate the basic ideas, consider the following numerical

Example: A regional health clinic intends to plan a schedule for its nurses. The estimated requirements are shown in Table 2.4.

Table 2.4

	Midnight – 4 a.m.	4 a.m. – 8 a.m.	8 a.m. – 12 noon	12 noon – 4 p.m.	4 p.m. – 8 p.m.	8 p.m. – midnight
Time slot #	1	2	3	4	5	6
Nurses needed	6	8	11	9	18	11

Defining a variable for each of the six time slots and assuming 8-hour shifts, we can formulate the problem as follows.

$$\text{Min } z = x_1 + x_2 + x_3 + x_4 + x_5 + x_6$$

$$\begin{aligned} \text{s.t.} \quad & x_1 + x_6 \geq 6 \\ & x_1 + x_2 \geq 8 \\ & x_2 + x_3 \geq 11 \\ & x_3 + x_4 \geq 9 \\ & x_4 + x_5 \geq 18 \\ & x_5 + x_6 \geq 11 \\ & x_1, \ldots, x_6 \geq 0. \end{aligned}$$

One optimal solution to this problem has $\mathbf{x} = [0, 8, 3, 6, 12, 6]$, so that a total of 35 nurses will be needed. In this solution, all constraints are tight, except for the 6th time slot, where we have an excess of seven nurses. Note that there are many alternative optimal solutions.

Suppose now that the second time slot requires 12, rather than 8, nurses, so that $b_2 = 12$. The new solution is then $\mathbf{x} = [1, 11, 0, 9, 9, 5]$, so that there is still a requirement for 35 nurses. Note that this solution is one of the many alternative optima for the original problem where it produced four excess nurses in time slot 2 and three excess nurses in time slot 6. Now the four excess nurses in time slot 2 are used up by the increased requirement (from 8 to 12), leaving the three excess nurses in time slot 6 as the only constraint that is not tight.

Various refinements of this basic model can be introduced. For instance, we can consider the assignments of individual employees to shifts. Appropriate variables can then be defined as x_{ij} which assume a value of one, if employee i is assigned to start work in shift j, and zero otherwise. We can then include constraints that require a minimal rest period between two shifts for each individual. However, this formulation is a model in integers which is beyond the scope of this volume. Instead, we refer interested readers to Eiselt and Sandblom (2000).

2.5 Data Envelopment Analysis

Data Envelopment Analysis (*DEA*) is an application of linear programming that deals with the comparison of different decision making units (*DME*s), such as branches of a fast-food chain, comparable regional health clinics, automobile dealerships selling the same brand, and similar units. The comparison is done with the help of input and output factors. In particular, for each unit (e.g., branch of a company) j, the method will construct a fictitious decision making unit as a linear combination of all other branches and compare its inputs and outputs with that of the j-th branch. If the fictitious unit can produce at least as much output with no more input than branch j, then branch j is not efficient. The active field of data envelopment analysis is based on the work by Charnes *et al.* (1978) who expanded upon ideas on efficiency put forward by Farrell (1957).

Formally, define the parameters $a_{i\ell}^{+}$ and $a_{k\ell}^{-}$ as the output of good i produced by branch ℓ and the input of factor k required by branch ℓ, respectively. Define then variables w_ℓ as the weights of branches $\ell = 1, \ldots$ that compose the fictitious unit, and denote the efficiency of branch j by E_j^D. The problem can then be formulated as follows.

P: Min E_j^D

$$\text{s.t.} \sum_\ell a_{i\ell}^+ w_\ell \geq a_{ij}^+ \ \forall \ i \tag{1}$$

$$\sum_\ell a_{k\ell}^- w_\ell \leq a_{kj}^- E_j^D \ \forall \ k \tag{2}$$

$$w_\ell \geq 0 \ \forall \ \ell, \ E_j^D \in \mathbb{R}.$$

The hypothetical branch mentioned above is constructed as a linear convex combination of the existing branches by means of the weights w_ℓ, $\ell = 1, \ldots, n$. The purpose is now to make this hypothetical branch as efficient (or even more efficient) than branch j, i.e., the branch whose efficiency is under investigation. This is accomplished by ensuring that the hypothetical unit produces as least as much output as unit j (via constraints (1)) and that the hypothetical unit uses no more input than unit j. The left-hand side of constraints (2) denotes the resource consumption of the hypothetical unit , whereas the right-hand side is the resource consumption of unit j weighted by its efficiency. For instance, if the efficiency of branch j is 0.8 (i.e., 80%), then constraints (2) guarantee that the hypothetical branch uses no more than 80% of the resources that are used by branch j, the branch whose efficiency is investigated. If the smallest value of E_j^D is 1, then the hypothetical branch cannot be more efficient as branch j which can then be considered (100%) efficient. If, on the other hand, the optimal value of $E_j^D = 0.8$, then the hypothetical branch can output as much as branch j with no more than 80% of its input; consequently, branch j can be said to be only 80% efficient.

Here we just state the basic problem of data envelopment analysis as shown above. In Section 8.3.4 we will demonstrate that the above problem P can be derived from some fundamental measure of efficiency.

The basic model as described here has many limitations. One such limitation is the difficulty of devising a quantitative measure for input and output factors. For instance, if hospitals are to be compared, the “quality of patient care” is certainly one of the more relevant features. It certainly is an output factor, but we will need some proxy expression in order to measure it. And therein lies the problem. Proxies never measure quite the same thing that they replace. For instance, measuring the quality of the care in terms of patient days spent in the hospital is not a good substitute for quality: long hospital stays may be safe for the patient, but they prevent patients to get on with their lives, not to mention those patients who are waiting for hospital beds to get the surgery they need. Much worse are other situations, e.g., when measuring the quality of educational institutions, often the amount of money spent on them, obviously an input factor, is used as output. Another serious shortcoming is the autocorrelation between different input and

output factors. For instance, sales, market share and revenue are all valid output factors, but they are obviously highly correlated.

Another shortcoming is the following. Suppose there is only a single input and a single output factor. There are three branches with outputs of 10, 6, and 8, respectively, while they require inputs of 8, 4, and 7, respectively. The third branch is not efficient, as a hypothetical branch that is a combination of the first two branches with weights 0.5 and 0.5 produces an output of 8, identical to that of branch 3, while it uses only 6 units of input, less than the 7 units required by branch 3. However, the hypothetical branch is just that—hypothetical. Maybe the decision maker finds that an output of 8 is exactly what is needed for his market niche and is satisfied with an input of 7.

Finally, note that data envelopment analysis will determine only relative efficiency in the sense that a branch will be considered efficient, whenever it compares favorably (by way of hypothetical units as explained above) with other existing branches. In case all branches perform terribly and one performs somewhat better than the rest, then the latter branch will be found "efficient."

Example: Consider four gas stations that belong to the same company and that are to be evaluated. The relevant input factors are the costs to run the station, the number of employees, and the square footage of the store that is attached to the station. The output factors are the revenues and the number of customers. The numerical information that was provided is shown in Table 2.5.

Table 2.5

	Costs (per day)	Employees	Square footage	Revenue (per day)	Customers (per day)
Branch 1	25	2	1,300	65	123
Branch 2	33	3	1,800	83	135
Branch 3	28	2	1,300	61	92
Branch 4	29	2	1,200	81	110

Notice the (at least partial) correlation between some of the factors, e.g., costs and the number of employees, and the number of customers per day and the revenue. For a proper analysis, it would be beneficial if such correlations were to be removed before the analysis commences, e.g., by way of cluster analysis that suggests to combine input or output criteria that are in some way correlated with each other.

Suppose now that we want to determine the efficiency of branch 3. The linear programming formulation of the problem is then as follows.

$$
\begin{aligned}
&\text{P: Min } E_3^D \\
&\text{s.t. } 65w_1 + 83w_2 + 61w_3 + 81w_4 \geq 61 \\
&\quad 123w_1 + 135w_2 + 92w_3 + 110w_4 \geq 92 \\
&\quad 25w_1 + 33w_2 + 28w_3 + 29w_4 \leq 28\,E_3^D \\
&\quad 2w_1 + 3w_2 + 2w_3 + 2w_4 \leq 2\,E_3^D \\
&\quad 1{,}300w_1 + 1{,}800w_2 + 1{,}300w_3 + 1{,}200w_4 \leq 1{,}300\,E_3^D \\
&\quad w_1, \quad w_2, \quad w_3, \quad w_4 \geq 0, \\
&\quad E_3^D \in \mathbb{R}
\end{aligned}
$$

Solving the problem, we find that $\overline{w}_1 = 0.2638$, $\overline{w}_2 = 0$, $\overline{w}_3 = 0$, $\overline{w}_4 = 0.5414$, and $\overline{E}_3^D = 0.8052$, indicating that branch 3 is only about 80% efficient. This means that a hypothetical branch that consists of $\dfrac{\overline{w}_1}{\sum_k \overline{w}_k} = 32.76\%$ of branch 1 and 67.24% of branch 4 will use only 80% of the input required by branch 4 to produce at least as much output as branch 3 does.

This is not really a surprise, as branch 3 is actually dominated by branch 1 in that it requires the same or more inputs than branch 1 and it produces lower output than branch 1. Some further details in the context of discrete data envelopment analysis are found in Eiselt and Sandblom (2004).

2.6 Inventory Planning

This section introduces a simple model that illustrates how inventories can be incorporated in a production problem similar to that introduced in Chapter 1 of this book. The basic reasons for inventories are manifold. The main reason is to avoid high costs of overtime, contracting out, and other stopgap measures in order to satisfy unexpected peaks in demand. The obvious drawback are the costs related to keeping inventories, such as warehouse operating costs, e.g., rent, heating/cooling, lighting, security, etc., but, most importantly, costs related to capital that is tied up in inventory. Hence, interest on capital tied up in inventories will make up most of the inventory costs.

By their very nature, problems involving inventories are dynamic. Each period, often a month, can be planned individually, and the inventory balancing constraints will serve as connecting links between the individual planning periods. Inventories can be visualized as holding tanks with the production serving as the inflow of goods and the demand as the outflow.

To formalize matters, first define d_t as the known demand in period t, where, for simplicity, we assume that the demand occurs at the end of the month. Next define the production variables x_t that denote the production of the good some time in period t, and define inventory variables I_t that denote the level of inventory at the beginning of period t. (Alternatively, we can define the inventory variables at the end of the period). We also have production capacities κ_t that may differ between the periods, production costs c_t^P that indicate the per-unit production costs in period t, and inventory holding costs c_t^I that express the cost of holding one unit in stock from the beginning of period t to the beginning of period t+1. The problem can then be formulated as

$$\text{P: Min } z = \sum_t c_t^P x_t + \sum_t c_t^I I_t$$

$$\begin{aligned} &\text{s.t. } x_t \le \kappa_t \ \forall\ t \\ &x_t + I_t \ge d_t \ \forall\ t \\ &I_{t+1} = I_t + x_t - d_t \ \forall\ t \\ &x_t, I_t \ge 0 \ \forall\ t. \end{aligned} \qquad (1)$$

The objective function minimizes the sum of production and inventory costs, while the constraints are the capacity constraints, the demand constraints (that require that the inventory available at the beginning of month t plus whatever is manufactured within month t is sufficient to satisfy the demand), and the typical inventory balancing constraints that specify that the inventory available at the beginning of month t+1 equals the inventory available at the beginning of the previous month t, plus whatever is produced in month t minus what is consumed (i.e., the demand) in month t. Clearly, all variables have to satisfy the usual nonnegativity constraints.

The problem can be simplified, though. Due to the nonnegativity constraints of the inventory constraints, we know that the inventory balancing constraints are $I_{t+1} = I_t + x_t - d_t \ge 0\ \forall\ t$ or, simply, $x_t + I_t \ge d_t\ \forall\ t$ as required by the demand constraints (1), making them redundant and therefore they can be deleted.

As an illustration, consider the following

Example: A company wants to plan its production for one of its products for the next four months. Table 2.6. shows the anticipated demand, the production capacities, and the unit production costs for the individual months, as well as the inventory holding costs that are incurred carrying over one unit from one month to the next.

Table 2.6

	Periods			
	Month 1	Month 2	Month 3	Month 4
Demand	50	120	150	160
Production capacity	100	100	160	150
Unit production cost	\$1	\$1.1	\$1.2	\$1.2
Inventory cost	\$.3	\$.2	\$.2	

At present, no units are in stock and after the four months, it is not desired to have any stock left. The problem can then be written as

$$\text{P: Min } z = 1x_1 + 1.1x_2 + 1.2x_3 + 1.2x_4 + .3I_2 + .2I_3 + .2I_4$$

$$\text{s.t. } x_1 \leq 100$$
$$x_2 \leq 100$$
$$x_3 \leq 160$$
$$x_4 \leq 150$$

$$I_1 = 0$$
$$I_2 = 0 + x_1 - 50$$
$$I_3 = I_2 + x_2 - 120$$
$$I_4 = I_3 + x_3 - 150$$
$$I_5 = 0 = I_4 + x_4 - 160$$

$$x_j, I_j \geq 0 \ \forall \ j$$

The optimal solution to this problem is $\mathbf{x} = [70, 100, 160, 150]$ and $\mathbf{I} = [0, 20, 0, 10]$ with associated costs of $\bar{z} = 560$.

As often, there are other ways of formulating the same problem. Rather than defining production and inventory variables separately as above, we can merge them and consider inventories implicitly. This is done by defining variables x_{ij} as the quantity that is made in period i and taken out of inventory in period j. Given this definition, inventory variables are no longer needed, but it is required to calculate the coefficients of the variables as $c_{ij} = c_i^P + \sum_{t=i}^{j-1} c_t^I$. The problem can then be formulated as

$$\text{P: Min } z = \sum_i \sum_{j \geq i} c_{ij} x_{ij}$$

$$\text{s.t. } \sum_{j \geq i} x_{ij} \leq \kappa_i \quad \forall \quad i$$

$$\sum_{i \le j} x_{ij} \ge d_j \quad \forall \quad j$$
$$x_{ij} \ge 0 \ \forall \ i, j.$$

It is noteworthy that given n planning periods, this formulation has n^2 decision variables as opposed to $2n$ variables in the formulation introduced in the beginning of this section. However, for a realistic number of planning periods, this should not pose a problem, given the efficiency of modern optimization codes. Furthermore, the latter formulation, while larger in size, does exhibit a very nice structure. In particular, its structure is one that equals that of so-called transportation problems that will be discussed in Section 2.8. In order to illustrate the concept, we will formulate the numerical example above again using this concept. This results in

$$\text{P: Min } z = 1x_{11} + 1.3x_{12} + 1.5x_{13} + 1.7x_{14} + 1.1x_{22} + 1.3x_{23} + 1.5x_{24} + 1.2x_{33} + 1.4x_{34} + 1.2x_{44}$$

$$\text{s.t.} \quad \left.\begin{aligned} x_{11} + x_{12} + x_{13} + x_{14} &\le 100 \\ x_{22} + x_{23} + x_{24} &\le 100 \\ x_{33} + x_{34} &\le 160 \\ x_{44} &\le 150 \end{aligned}\right\} \text{ production capacities}$$

$$\left.\begin{aligned} x_{11} &\ge 50 \\ x_{12} + x_{22} &\ge 120 \\ x_{13} + x_{23} + x_{33} &\ge 150 \\ x_{14} + x_{24} + x_{34} + x_{44} &\ge 160 \end{aligned}\right\} \text{ demand constraints}$$

$$x_{ij} \ge 0 \ \forall \ i, j.$$

The optimal solution in terms of the optimal values of the variables x_{ij} is shown in Table 2.7, where the rows refer to the production period, while the columns refer to the period, in which the units are taken out of stock.

Table 2.7

	Period 1	Period 2	Period 3	Period 4
Period 1	50	20	0	0
Period 2	–	100	0	0
Period 3	–	–	150	10
Period 4	–	–	–	150

The total cost at optimum are 560. Note that the production quantities in the individual periods are the sums of production quantities in the rows in the matrix above. For instance, in Period 1 we make 50 + 20 + 0 + 0 = 70 units.

In this formulation, it is very useful to draw up a time line in order to visualize the in- and outflows into the inventory. For the example above, such a time line would look as shown in Figure 2.3

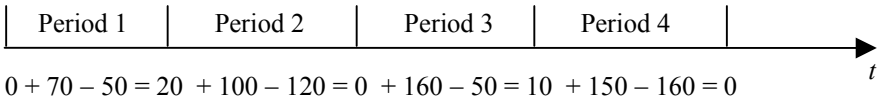


Figure 2.3

Here, it becomes apparent that the inventory levels at the beginning of the four periods are 0, 20, 0, and 10, respectively. This is exactly the same solution as that obtained by the other formulation above.

2.7 Blending Problems

Blending problems have a long history in the applications of linear programming. One of the first descriptions of blending problems deals with the blending of gasolines, see Charnes *et al*. (1952). Their paper describes a linear programming problem that blends airline fuels and adds chemicals, so as to ensure that prespecified performance levels are attained, e.g., vapor pressure, lead and sulfur content and other specifications. The objective function is to maximize the profit. Other popular examples of blending problems comprise tees, coffees, tobacco, or similar products. The general structure can be described as follows. One set of subscripts includes all of the m available raw materials, while another includes all of the n desired final products. As usual, only limited quantities of the raw materials are available, and certain amounts of the final products have to be blended. Here, we assume that raw materials blend linearly, meaning that taking, say, α units of raw material A and β units of raw material B, then the resulting blend C has features that are proportional to the quantities of A and B that C is made of. As an example, take 3 gallons of 80º water and 2 gallons of 100º water, then the result would be 5 gallons of water, whose temperature is [3(80) + 2(100)]/5 = 88º. Assume now that we have a supply of s_i units of the i-the raw material while d_j units of the j-the product are in demand. The unit costs of the i-th raw material is c_i, whereas the unit price of the j-th final product is p_j. In order to ensure a consistent quality of the blend, some restrictions apply. The parameters $\underline{a}_{ij}$ and $\overline{a}_{ij}$ indicate the smallest and largest proportion of raw material i that is allowed in the final product j. For instance, if the two values are 0.3 and 0.5, respectively, then the content of raw material i in product j must be at least 30%

and cannot exceed 50%. It is apparent that for cheaper products, the range between $\underline{a}_{ij}$ and $\overline{a}_{ij}$ will typically be allowed to be quite large, while high-quality products will have to be manufactured within very tight tolerances.

In order to formulate the problem, define variables x_{ij} that denote the quantity of raw material i in product j. Then we obtain the problem

$$\text{P: Max } z = \sum_j p_j \sum_i x_{ij} - \sum_i c_i \sum_j x_{ij}$$

$$\text{s.t. } \sum_j x_{ij} \le s_i \quad \forall \ i$$

$$\sum_i x_{ij} = d_j \quad \forall \ j$$

$$x_{ij} \ge \underline{a}_{ij} \sum_k x_{kj} \quad \forall \ i,j$$

$$x_{ij} \le \overline{a}_{ij} \sum_k x_{kj} \quad \forall \ i,j$$

$$x_{ij} \ge 0 \ \forall \ i,j.$$

As an illustration, consider the following

Example: A firm faces the problem of blending three raw materials into two final products. The required numerical information is provided in Table 2.8.

Table 2.8

Products / Raw Materials	1	2	Amount available (in tons)	Unit Cost ($)
1	[.4; .6]	[.5; .6]	2,000	1.00
2	[.1; .2]	[.1; .4]	1,000	1.50
3	[.2; .5]	[.2; .3]	500	3.00
Quantity required in tons	600	700		
Unit selling price ($)	10	8		

Define variables x_{ij} as the quantity of raw material i in product j. The formulation is then

$$\text{P: Max } z = 10(x_{11} + x_{21} + x_{31}) + 8(x_{12} + x_{22} + x_{32}) - [1(x_{11} + x_{12}) + 1.5\,(x_{21} + x_{22}) + 3(x_{31} + x_{32})]$$

$$\text{s.t. } x_{11} + x_{12} \le 2{,}000$$
$$x_{21} + x_{22} \le 1{,}000$$

$$x_{31} + x_{32} \leq 500$$

$$x_{11} + x_{21} + x_{31} = 600$$
$$x_{12} + x_{22} + x_{32} = 700$$

$$x_{11} \geq 0.4(x_{11} + x_{21} + x_{31}) \qquad x_{11} \leq 0.6(x_{11} + x_{21} + x_{31})$$
$$x_{21} \geq 0.1(x_{11} + x_{21} + x_{31}) \qquad x_{21} \leq 0.2(x_{11} + x_{21} + x_{31})$$
$$x_{31} \geq 0.2(x_{11} + x_{21} + x_{31}) \qquad x_{31} \leq 0.5(x_{11} + x_{21} + x_{31})$$
$$x_{12} \geq 0.5(x_{12} + x_{22} + x_{32}) \qquad x_{12} \leq 0.6(x_{12} + x_{22} + x_{32})$$
$$x_{22} \geq 0.1(x_{12} + x_{22} + x_{32}) \qquad x_{22} \leq 0.4(x_{12} + x_{22} + x_{32})$$
$$x_{32} \geq 0.2(x_{12} + x_{22} + x_{32}) \qquad x_{32} \leq 0.3(x_{12} + x_{22} + x_{32})$$

$$x_{ij} \geq 0 \ \forall \ i, j.$$

The optimal solution includes the quantities shown in the following matrix, whose components are the optimal values of the variables x_{ij}. The transportation plan is

$$\begin{bmatrix} 360 & 420 \\ 120 & 140 \\ 120 & 140 \end{bmatrix}$$

and the profit is \$9,650. It will be interesting—albeit predictable—to see what happens if the blending requirements are tightened. In particular, reduce the upper bounds of the ranges relating to product 1 from .6, .2, and .5 to .5, .1, and .4. The new optimal solution then includes the quantities

$$\begin{bmatrix} 300 & 420 \\ 60 & 140 \\ 240 & 140 \end{bmatrix}$$

with a total profit of \$9,440. In other words, the more tightly controlled product 1 causes rather significant changes in the blending schedule and reduces the profit by \$210 or 2.2%, i.e., by 35¢ for each unit of product 1 that is sold. If the buyers are prepared to pay that much more for an improved product 1 with tighter quality controls, this is certainly an option to consider.

2.8 Transportation Problems

Transportation costs are a significant expense in the total cost of the product that range on average at about 5% of the total price of manufactured products. It is therefore imperative that the planner attempt to keep these costs at as low a level as possible.

Attempts of this nature are certainly not new. As a matter of fact, Hitchcock's (1941) work predates the emergence of linear programming, but was later shown to be a special case of it.

In this section, we will refer to "the" transportation problem as a specific structure while we acknowledge that it is but one of a large number of transportation scenarios that may apply in any given situation. The basic structure of the transportation problem is as follows. On the one hand we have *origins* O_i, i=1, …, m at which supplies s_i of a single product are available. On the other side are *destinations* D_j, j=1, …, n at which customers demand goods in quantities of d_j. Shipments of the good are possible only directly from an origin to a destination, detours are not permitted and deliveries cannot be combined. Transporting a single unit from origin O_i to destination D_j is assumed to cost c_{ij}. The transportation costs are assumed to be linear. This situation can be visualized as shown in Figure 2.4.

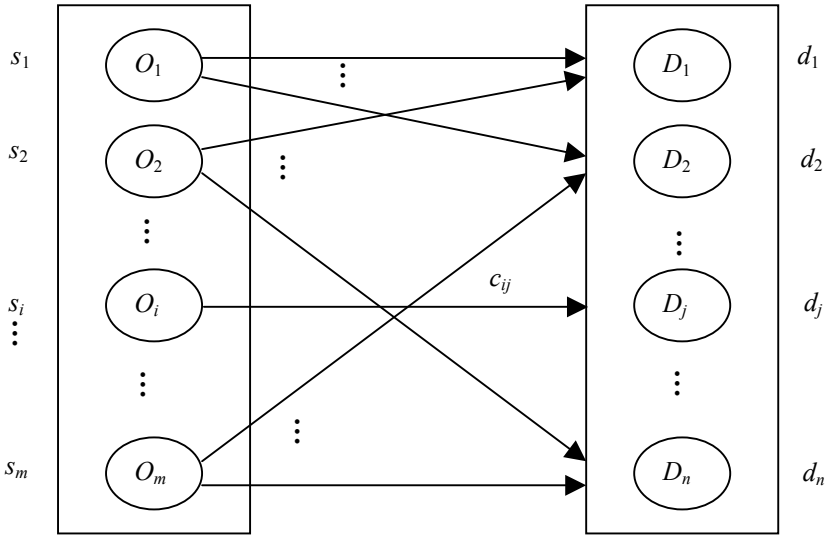


Figure 2.4

While the assumption of known supply is reasonable, the assumption of known demand may be justified in instances, in which the system fulfills given orders. The lack of economies of scale that is required to justify linear transportation costs is somewhat harder to defend. Even in the absence of nonlinearities, transportation cost functions are typically nonlinear. Consider the case of simple deliveries. The first case is the transportation of, say, snowblowers with a pickup truck. Assume that a single trip from the warehouse to the customer costs $20, then the shipment of no snowblowers costs $0, hauling one snowblower (requiring one trip) costs $20, hauling two snowblowers (requiring two separate trips) costs $40, and so forth. The result is a function that is defined only for integers, but if these were

connected, this would be a linear function. The second scenario is the shipment of small boxes by pickup truck. Again, no deliveries cost \$0 and hauling a single box requires one trip and hence costs \$20. However, transportation of the second box costs nothing extra (other than loading and unloading fees). The same argument applies to the third, fourth box, and so forth. However, once the capacity κ of the truck has been reached, a second trip will be required, so that the costs jump by \$20. The resulting piecewise linear step function is shown in Figure 2.5 The broken line in this figure indicates a linear function that may approximate the step function, which is very awkward to handle. Whether the linear cost function in our transportation problem is originally linear as in the case of the snowblowers or is an approximation as in the case of the boxes, is irrelevant for its application.

For now, we will also assume that the transportation is *balanced*, i.e., the total supply equals the total demand. More formally, we require that $\sum_{i=1}^{m} s_i = \sum_{j=1}^{n} d_j$. With the assumptions above, we now wish to minimize the total transportation costs, given that all demands are satisfied. In order to formulate the problem, we first define variables x_{ij} as the quantity shipped (directly) from origin O_i to destination D_j. The problem can then be written as

$$\text{Min } z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$$

$$\text{s.t. } \sum_{j=1}^{n} x_{ij} = s_i \quad \forall\ i = 1,...,m$$

$$\sum_{i=1}^{m} x_{ij} = d_j \quad \forall\ j = 1,...,n$$

$$x_{ij} \geq 0 \qquad \forall\ i=1, \ldots, m; j=1, \ldots, n.$$

It should be noted that the standard transportation problem as shown in the above formulation has mn variables and $(m+n)$ structural constraints. However, one constraint is redundant, which can easily be seen by adding the first m constraints as well as the next n constraints. The results will be the total supply and demand, respectively. The assumption of a balanced problem then requires these two to be equal, so that any one of the $(m+n)$ constraints can be expressed in terms of the remaining $(m+n-1)$ constraints. In other words, the matrix of constraints does not have full rank and any basis (see Definition A.17) will include $(m+n-1)$ rows and columns. This issue is further explored in Chapter 3, where the simplex method is described.

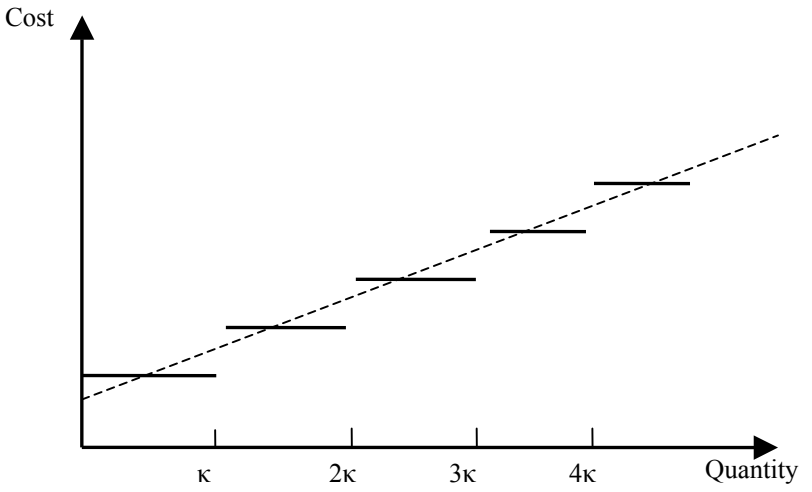


Figure 2.5

Example: Consider a problem with three origins and four destinations. The supplies at the origins are 30, 60 and 50, respectively, while the demands at the destinations are 20, 40, 50, and 30, respectively. Note that as the total supply and the total demand both equal 140, the transportation problem is balanced. Let the unit transportation costs be shown in the cost matrix

$$\mathbf{C} = \begin{bmatrix} 5 & 6 & 4 & 5 \\ 3 & 4 & 3 & 2 \\ 2 & 7 & 6 & 8 \end{bmatrix}.$$

The problem can then be formulated as follows.

P: Min $z = 5x_{11}+6x_{12}+4x_{13}+5x_{14}+3x_{21}+4x_{22}+3x_{23}+2x_{24}+2x_{31}+7x_{32}+6x_{33}+8x_{34}$

$$\begin{aligned}
\text{s.t. } & x_{11} + x_{12} + x_{13} + x_{14} = 30 \\
& x_{21} + x_{22} + x_{23} + x_{24} = 60 \\
& x_{31} + x_{32} + x_{33} + x_{34} = 50 \\
& x_{11} + x_{21} + x_{31} = 20 \\
& x_{12} + x_{22} + x_{32} = 50 \\
& x_{13} + x_{23} + x_{33} = 40 \\
& x_{14} + x_{24} + x_{34} = 30 \\
& x_{11},\ x_{12},\ x_{13},\ x_{14},\ x_{21},\ x_{22},\ x_{23},\ x_{24},\ x_{31}\ x_{32},\ x_{33},\ x_{34} \geq 0
\end{aligned}$$

Notice the special (block-angular) structure. It is shown again in Figure 2.6 where the shaded areas are the parts in the formulation that have coefficients of "1", while the remaining coefficients are all zero.

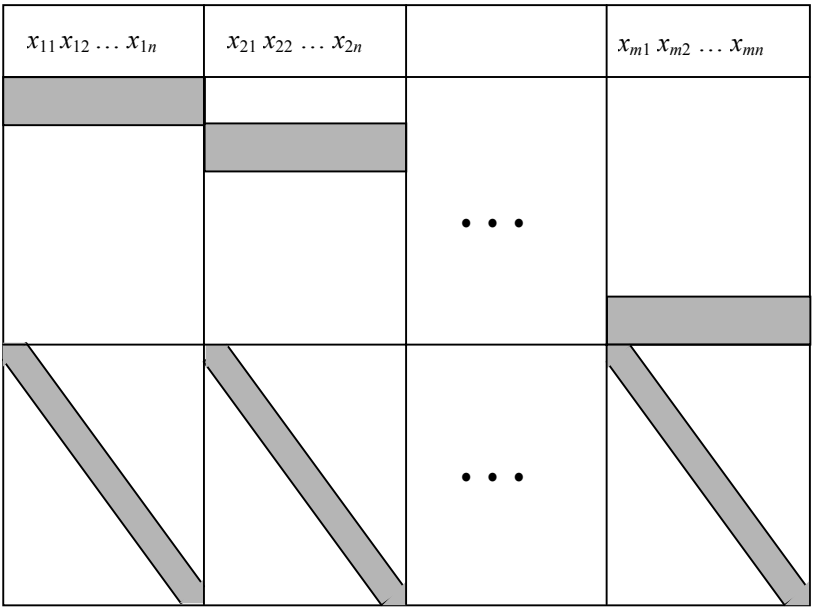


Figure 2.6

This structure does two things: first it allows the design of very fast specialized algorithms. Given the advances in software codes, this issue is no longer as relevant as it was twenty or thirty years ago. Very large transportation problems can be solved today without difficulty, so that we have decided not to describe any of those specialized techniques, commonly referred to as modified distribution (MODI = modified distribution technique) and/or stepping stone method. For a detailed description of the method, see, e.g., Dantzig and Thapa (1997, 2003) or Eiselt *et al*. (1987). Secondly, the special structure implies that the matrix of coefficients is totally unimodular, meaning that each of its square nonsingular submatrices has a determinant of +1 or −1. By way of Cramer's rule (see Procedure A.18), this implies that whenever all right-hand side values, i.e., the supplies and demands, are integers, then at least one optimal solution will also be integer.

The solution of transportation problems can be displayed in a figure similar to that of Figure 2.4. However, with increasing problem size this gets very messy and it is preferable to display the solution in the form of a transportation plan T, which is a matrix whose element (i, j) indicates the number of units shipped from origin O_i to destination D_j, i.e., the optimal value of the variable x_{ij}. The optimal trans-

portation plan of the above problem is

$$T = \begin{bmatrix} 0 & 0 & 30 & 0 \\ 0 & 30 & 0 & 30 \\ 20 & 10 & 20 & 0 \end{bmatrix}$$

and the associated value of the objective function is $\bar{z} = 530$.

Consider now a transportation problem that is not balanced. In such a case, the above formulation can no longer be used as it includes a contradiction: The first n constraints force all available supplies out of the origins and into the network, while the second m constraints force all units in the network to the destinations. If, as is the case in all unbalanced problems, those two quantities are not equal, no feasible solution will exist because of this contradiction.

Suppose first that the total supplies are less than the total demand. This means that the total quantity that is hauled out of the origins and into the network is less than what the customers would want. This can be modeled by keeping the supply constraints as equations, while writing the demand constraints as ≤ inequalities. In this case, some demand will not be met. On the other hand, if more supplies are available than are demanded by the customers, not all supplies can be shipped out of the origins, as it will not be accepted at the destinations. Hence, in such a case the supply constraints will be written as ≤ inequalities, while the demand constraints may remain as equations.

Consider now the following linear programming problem:

$$\begin{aligned} \text{P: Min } z &= 5x_1 + 3x_2 \\ \text{s.t.} \quad x_1 &\leq 10 \\ x_2 &\leq 20 \\ x_1 + x_2 &= 25 \\ x_1, \quad x_2 &\geq 0. \end{aligned}$$

This could be the formulation of a transportation problem with two origins and one warehouse, where the supplies are 10 and 20 units, respectively, while the demand is 25, and the unit transportation costs from the origins to the destination are \$5 and \$3, respectively. However, the very same problem could alternatively model a blending problem with two raw materials and one finished product with appropriate supplies and demand as well as blending costs of 5 and 3, respectively. The important point here is that the mathematical optimizer, i.e., the software, only knows the formulation above and not what is behind it. Hence it will not be able to eliminate solutions that may make sense for one problem but not the other.

There are a number of formulations other than blending problems that have apparently nothing to do with transportation, yet can be reduced to the same structure. The second formulation in Section 2.6 on Inventory Planning is one such example.

Another such example concerns the assignment of workers to shifts. The simplest version of this worker-job assignment can be described by the following

Example: The set of workers can be divided into different groups with wages w_1, w_2, w_3, and w_4, respectively. These classifications may be based on abilities, seniority, or a combination of both or any other factors deemed relevant. Suppose that the hourly base wages are \$8, \$11, \$13, and \$16, respectively. We assume that all workers have the ability to perform any of the tasks in question. Assume that the four wage groups comprise 10, 20, 15, and 15 workers, respectively. The tasks at hand have to be performed in three shifts, day, evening, and night. In order to ensure continuity in production, 20 workers are required in each shift. The day shift pays base salaries, the evening shift pays a premium of 20%, and the night shift pays a premium of 50%. Denoting by p_i the hourly wage of group i and by q_j the multiplier applied to shift j, a member of the i-th group working in shift j will then be paid $c_{ij} = p_i q_j$. Eiselt and Gerchak (1984) have referred to such cost-coefficients as "scalar-generated." Applied to our example, the cost matrix is then

$$\mathbf{C} = \begin{bmatrix} 8.00 & 9.60 & 12.00 \\ 11.00 & 13.20 & 16.50 \\ 13.00 & 15.60 & 19.50 \\ 16.00 & 19.20 & 24.00 \end{bmatrix}$$

Define variables x_{ij} as the number of workers in wage group i that are assigned to work in shift j, and assume that the problem is to assign workers to shifts so as to minimize the total costs. It then becomes apparent that the problem is nothing but a transportation problem (or, rather, has the same structure as the standard transportation problem). Here, the sum of assignments per wage group must equal (or, at least, not exceed) the number of workers in that group and the sum of assignments in each shift must equal (or, at least, be no less) than the number of workers required in that shift. In our example, the problem happens to be balanced.

The solution of this problem is particularly simple on account of the special structure of the cost coefficients. As a matter of fact, a so-called "northeast corner rule" can be shown to always find at least one optimal solution. This rule will assign as many workers to the element in the northeast corner (here 10, as there are only 10 workers available in that wage group). Formally, we set $\bar{x}_{13} = 10$. Since this assignment leaves no workers in the first wage group, we must also set $\bar{x}_{12} = \bar{x}_{11} = 0$. Furthermore, we need to update the number of required workers in

the third shift which is now only 10, as 10 workers have just been assigned to that shift. The next unassigned element in the northeast corner is x_{23}. The second wage group still has its original 20 members unassigned, while the third shift only requires 10 more workers, so that we will set $\bar{x}_{23} = 10$. Again, we update the number of workers in wage group 2 that are still available (which is now10) as well as the number of additional workers required in the third shift (which is now 0). In order to ensure that no further workers are assigned to the third shift, we will set $\bar{x}_{33} = \bar{x}_{43} = 0$. The northwest corner is now at wage group 2 and shift 2. In this wage group, there are still 10 workers unassigned, while 20 are needed in shift 2. Consequently, we will assign $\bar{x}_{22}$ = 10 workers and update the number of unassigned workers in wage group 2 (which is now 0) and the number of workers still missing in shift 2 (which is now 10). In order to avoid assigning more workers of wage group 2 to other assignments, we will set $\bar{x}_{21} = 0$. The northwest corner is now reached at element (3, 2), for which 15 workers of wage group 3 are available, while 10 workers are still needed in shift 2. We assign $\bar{x}_{32}$ = 10 workers and by setting $\bar{x}_{42} = 0$, we avoid assigning more workers to shift 2 than are actually needed. Updating supply and demand results in 5 workers still unassigned in wage group 3, while no more workers are needed in shift 2. The remaining assignments are now uniquely determined, so that we can set $\bar{x}_{31} = 5$ and $\bar{x}_{41} = 15$.

An optimal solution is therefore

$$\mathbf{T} = \begin{bmatrix} 0 & 0 & 10 \\ 0 & 10 & 10 \\ 5 & 10 & 0 \\ 15 & 0 & 0 \end{bmatrix} \text{ with costs } \bar{z} = 878 .$$

The fact that the northeast rule results in an optimal solution implies that an optimal solution will assign as many members as possible of the lowest wage group to the highest paying shift (here: the night shift) while it will assign as many members as possible of the highest wage group to the lowest paying shift (here: the day shift). Actually, this assignment remains optimal as long as (a) night shift pays not less than evening-shift and evening-shift pays no less than day shift and (b) a member of a higher wage group is not paid less than a member of a lower wage group. Both assumptions are highly unlikely to be violated. Therefore the solution for this type of model is extremely stable. It should be noted, however, that it is not too realistic to consistently assign members of lower wage groups to night shifts. On the other hand, an optimal solution for the above problem will also result in relatively small differences in the wages (since low-paid workers are preferably assigned to the higher-paying shifts and high-paid workers preferably to the low-paying shifts).

Transportation problems have a variety of extensions. In case capacities are imposed upon the transportation links, appropriate capacity constraints $x_{ij} \leq \kappa_{ij} \; \forall \; i, j$ with capacities κ_{ij} can easily be added to the formulation. Another extension includes *transshipment points*. Such points may facilitate the transfer of goods from large trucks used for long-distance hauls to smaller local delivery trucks in a multimodal transportation system.

Similarly, transportation systems that allow shipments on road and rail will use transshipment points, as does municipal solid waste at transfer points. The existence of such transshipment points does, surprisingly enough, not destroy the structure of the transportation problem as introduced above. As a matter of fact, transshipment problems can be reformulated, so that they appear exactly like transportation problems.

This is accomplished by the inclusion of a preprocessing phase in the algorithm, in which the shortest paths between all origins and destinations are determined, and the unit transportation costs then denote the sum of unit costs that are incurred on that path. The transportation problem is then solved disregarding the transshipment points. A shipment of x_{ij} units from origin O_i to destination D_j must then be interpreted as a shipment of x_{ij} units on *all* connections on the path from O_i to D_j. For further details, readers are referred to Eiselt and Sandblom (2000).

Another extension concerns capacities associated with the routes. Again, it is possible to reformulate capacitated transportation problems as regular transportation problems, albeit at the expense of a fairly dramatic increase in size.

Much more difficult is the relaxation of the assumption that each unit of the products is shipped on a single trip. If trips are combined, the result is a routing problem. Such problems are typically **NP**-hard. Some routing problems are discussed in Eiselt and Sandblom (2000).

Consider now the assumption that goods have to be shipped directly from origin to destination without rerouting. Relaxing this assumption results in a transportation problem with *reshipments*. In simple words, reshipments allow goods to be shipped back and forth between origins and destinations so as to avoid the use of costly direct connections. Transportation problems with reshipments were first introduced by Dwyer (1975) and have subsequently been discussed by Finke (1977)

In order to solve reshipment problems, we define again variables x_{ij} as the quantity shipped from origin O_i to destination D_j, where a negative value of a variable indicates that the shipment is directed from a destination to an origin. The problem can then be formulated as follows:

$$\text{P: Min } z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} \mid x_{ij} \mid$$

$$\text{s.t. } \sum_{j=1}^{n} x_{ij} = s_i \quad \forall\ i = 1, ..., m$$

$$\sum_{i=1}^{m} x_{ij} = d_j \quad \forall\ j = 1, ..., n$$

$$x_{ij} \in \mathbb{R} \qquad \forall\ i=1, \ldots, m; j=1, \ldots, n.$$

It is apparent that the only difference of the transportation problem with reshipments and the standard problem is the existence of unrestricted variables in the former and the use of absolute values in the objective function. Both features can be transformed, so that the problem appears again as a standard linear programming problem. For details concerning this type of transformation, see Section 8.3.1 of this volume.

As an illustration, consider the following

Example: Supplies of 10 and 10 units are available at the two origins, while the demands at the two destinations are 5 and 15, respectively. The unit transportation costs are summarized in the cost matrix $\mathbf{C} = \begin{bmatrix} 2 & 6 \\ 1 & 2 \end{bmatrix}$. The optimal transportation plan without reshipments is $T = \begin{bmatrix} 5 & 5 \\ 0 & 10 \end{bmatrix}$, while the optimal transportation plan with reshipments is $T = \begin{bmatrix} 10 & 0 \\ -5 & 15 \end{bmatrix}$. The latter plan clearly shows a reshipment from destination D_1 back to origin O_2. The interesting is though, that the optimal solution of the standard problem costs \$60, while reshipments reduced these costs to \$55. The reason for this cost reduction become apparent upon closer inspection. While the solution of the standard transportation problem ships 5 units on the very expensive route from O_1 to D_2, the reshipment version does not (thus saving \$6 per unit). Instead, it ships 5 additional units from O_1 to D_1 (at additional costs of \$2 per unit), hauls them back from D_1 to O_2 (at additional costs of \$1 per unit), and finally ships them from O_2 to their final destination at D_2 (at an additional cost of \$2 per unit). Adding up the additional costs yields \$5 per unit, which must be contrasted with savings of \$6 per unit, resulting in net savings of \$1 per unit. Since 5 units were rerouted in this fashion, the savings are \$5, which explains the decrease of the total costs from \$60 to \$55.

Finke (1983) has outlined conditions under which reshipments yield savings beyond the optimal solution of the standard transportation problem. The simple existence conditions derived by the author require dual variables as discussed in

Chapter 4 in this volume. The solutions of randomly generated test problems could be improved significantly by allowing reshipments.

Another extension of the standard transportation problem is referred to as a problem with *overshipments*. Simply speaking, it allows additional units to flow through the network beyond those presently available at the origins and those demanded at the destinations. Surprisingly, such overshipments may actually result in cost savings, something dubbed the *more-for-less paradox*. The paradox is due to Swarc (1971), Charnes and Klingman (1971). The formulation of the problem is very simple: all constraints are of the type "≥", as it is allowed to ship more units than currently available, and the customers are assumed to be willing to accept more units than they originally asked for.
As a numerical illustration of overshipments, consider the following

Example: Supplies of 10 units are available at each of the two origins, while demands at the two destinations are 15 and 5 units, respectively. The unit transportations costs are summarized in the cost matrix $\mathbf{C} = \begin{bmatrix} 1 & 4 \\ 4 & 1 \end{bmatrix}$. The optimal transportation plan of the standard problem is then $\overline{T} = \begin{bmatrix} 10 & 0 \\ 5 & 5 \end{bmatrix}$ with a value of the objective function of $\bar{z} = 35$. Suppose now that is were possible to obtain an additional unit of the good at origin O_1 and to convince the customer at destination D_2 to accept an extra unit of the product. Solving the new transportation problem results in an optimal transportation plan $\overline{T}' = \begin{bmatrix} 11 & 0 \\ 4 & 6 \end{bmatrix}$ whose costs are now only $\bar{z}' = 33$.

The reason for the savings is the same as in case of reshipments: replacing the use of expensive links by cheaper connections. In this example, one less unit is shipped from origin O_2 to destination D_1, a move that results in savings of \$4. Instead, additional units are shipped from O_1 to D_1 (additional cost: \$1) and from O_2 to D_2 (additional cost: \$1) for net savings of \$2, which explains the reduction of total costs from \$35 to \$33. So again, there are potential savings beyond the standard transportation solution. However, this case is much more difficult than that of reshipments: in this case, additional supplies have to be obtained and customers have to be convinced (presumably by using monetary incentives) to accept larger quantities than originally planned. Again, existence conditions based on duality theory are available, see Finke (1977).

Sensitivity analyses on transportation problems can be performed similar to those applied to standard linear programming. Clearly, if we start with a balanced transportation problem and change a single right-hand side value, the problem will become unbalanced, thus potentially requiring a modified formulation. Similarly, whenever the change changes the balance (i.e., changing a net demand surplus to a

net supply surplus or vice versa), the formulation will have to change. In those cases it is probably easiest to resolve the problem.

It must be stressed that it is crucial to apply the classical transportation problem only to scenarios that satisfy all assumptions, most prominently the assumption that we are dealing with a homogeneous commodity that is transported. This means that, for instance, the classical transportation model is not suitable to deal with, say, traffic problems which are highly heterogeneous, in that each origin-destination pair represents one individual and thus one "commodity."

Furthermore, most transportation scenarios are, of course, more complex than the simple structure described here. So in general, it is best to put models that deal with transportation in the context of network flow models. Readers are referred to Eiselt and Sandblom (2000), Ahuja *et al*. (1993), or Murty (1992).

2.9 Assignment Problems

The problem described in this section is a close relative of the Transportation Problem discussed in the previous section. It can briefly be described as follows. Suppose there are n employees and n tasks. The goal is now to assign employees to tasks, so that each employee works on exactly one task, each task is worked on by exactly one employee, and the cost of assigning employees to tasks is minimized. Early work on the problem started with Egerváry's (1931) combinatorial theorem and Kuhn's (1955) "Hungarian algorithm," so called in honor of Egerváry's contribution.

A story that was used to sell the problem soon after its appearance in the 1950s is the "Marriage Problem." The idea is to match the offspring of a family, e.g., the daughters, and marry them to a set of eligible bachelors. Each match requires a certain amount of dowry to be paid, and the objective of the brides' father, who presumably foots the bill, is to minimize the total amount of dowry. If it were permitted that the individuals spend partial time with the various partners of the other sex, an assignment problem results of the type described above. Using the above arguments, it can then be shown that monogamy i.e., the fact that at least one optimal solution is in integers, i.e., has assignments in which partners do not share time, is optimal. It remains optimal, even if the central planner changes the objective from cost minimization to the maximization of overall happiness (or compatibility), an objective, whose coefficients c_{ij} measure the degree of happiness if girl i is matched with boy j.

This is the problem statement found in most textbooks. Here, we would like to start the discussion with a slightly different description. In particular, we assume that all employees, once properly trained, work on tasks at about the same speed. All employees start with the same amount of time (e.g., one hour, one day, or a similar measure), and each task requires the full amount of time each employee

has available (or the equivalent of multiple employees). The cost of assigning an employee to a job includes the (re-) training cost required to enable to employee to properly work on the task. The employer's task is then to minimize the total training costs.

In order to formulate this problem, define variables x_{ij} as the proportion of employee i's time that is allocated to task j. Given c_{ij} as the cost to train employee for task j, we can write the problem as follows.

$$\text{P: Min } z = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

$$\text{s.t. } \sum_{i=1}^{n} x_{ij} = 1 \;\; \forall \;\; j = 1, ..., n$$

$$\sum_{j=1}^{n} x_{ij} = 1 \;\; \forall \;\; i = 1, ..., n$$

$$x_{ij} \geq 0 \; \forall \; i, j = 1, \ldots, n.$$

It is apparent that the assignment problem is nothing but a transportation problem with all supplies and demands equal to one. However, not only can an assignment problem be seen as a special case of a transportation problem, it is also possible to formulate each balanced transportation problem as an assignment problem. This can be accomplished by splitting up the supply s_i (and, similarly, the demand d_j) at origin i (destination j) into s_i (d_j) supply (demand) points, each with a demand of one. That way, all supplies and demands equal one, which is the situation in the assignment problem. There is a tremendous cost, though, in terms of a greatly increased problem size. Unbalanced problems are dealt with similarly as in transportation problems.

Since the structure of assignment problems is identical to that of transportation problem (only the right-hand side values differ), their properties are also the same. In particular, the matrix of coefficients is totally unimodular, so that for integer right-hand side values (and in assignment problems they are naturally integer), each solution will be integer. This implies that all variables will assume values of zero or one, since if a variable had a larger integer value, another variable would have to be negative which is prohibited by the nonnegativity constraints. This, in turn, implies that each employee will be assigned to work on exactly one task and each task will be performed by exactly one employee. This is, of course, the original problem definition at the beginning of this section. The difference here is that we have started with a more general problem statement and the "one employee for one task" property is not a requirement formulated in the constraints but a feature that happens to occur.

If some assignments are prohibited, e.g., an employee is not allowed to work on a task (e.g., due to inability, lack of security clearance, or similar reasons), the assignment variable should assume a value of zero. The easiest way to ensure that this is satisfied is to simply not include the variable in the problem.

In addition to the aforementioned employee-to-task assignments, there are various other applications of assignment problems. Assigning police patrol cars to districts (with the costs indicating the effectiveness of a certain team to work a district), assigning salesmen to districts (with the likely sales as cost coefficients), and similar assignments suggest themselves. Another popular application concerns an assignment of athletes to teams. Here, we will consider eight boxers from two teams that are supposed to be paired up for an upcoming event. The two teams include fighters F_1, F_2, F_3, and F_4, and their opponents are O_1, O_2, O_3, and O_4. Clearly, the way the boxers are teamed up will determine the attractiveness of the event and with it the revenue. It has been estimated that the attractiveness of a pairing (F_i, O_j) can be measured by an exponential function $50{,}000\, e^{-(|f_i - o_j|+1)}$, where f_i and o_j are the abilities of fighters F_i and O_j, respectively. However, we cap the attractiveness at 10,000. (The idea here is that pairing fighters with similar abilities will result in more interesting fights). The promoter's objective is then to maximize the overall attractiveness of the pairings. Consider the following numerical

Example: The first teams consists of four fighters with abilities 7, 3, 6, and 5, respectively, while the opposing team has individuals with abilities 8, 2, 4, and 5. The matrix

$$\mathbf{M} = \begin{bmatrix} 6{,}766 & 124 & 916 & 2{,}489 \\ 124 & 6{,}766 & 6{,}766 & 2{,}489 \\ 2{,}489 & 337 & 2{,}489 & 6{,}766 \\ 916 & 916 & 6{,}766 & 10{,}000 \end{bmatrix}$$

includes the utilities for all pairings (F_i, O_j). Solving the assignment problem (with maximization objective) results in the pairings (F_1, O_1), (F_2, O_2), (F_3, O_4), and (F_4, O_3) with a total attractiveness of $\bar{z} = 27{,}064$.

In order to compute the "value added" of each fighter, we can now proceed as follows. We first calculate the total attractiveness without a single fighter F_i, or O_j, for all i and j. For fighter F_k, this problem is identical to the previous problem, except that we set the variables $x_{kj} := 0 \; \forall \, j$ and write the first set of constraints as $\sum_{i=1}^{n} x_{ij} \leq 1 \;\; \forall \;\; j = 1, ..., n$. In our example, calculating the value of fighter F_1 in the event is determined by solving the problem

$$\text{P: Max } z_{F_i} = 124x_{21} + 6{,}766x_{22} + 6{,}766x_{23} + 2{,}489x_{24} + 2{,}489x_{31} + 337x_{32} + 2{,}489x_{33} + 6{,}766x_{34} + 916x_{41} + 916x_{42} + 6{,}766x_{43} + 10{,}000x_{44}$$

$$\begin{aligned}
\text{s.t.} \quad & x_{21} + x_{22} + x_{23} + x_{24} = 1 \\
& x_{31} + x_{32} + x_{33} + x_{34} = 1 \\
& x_{41} + x_{42} + x_{43} + x_{44} = 1 \\
& x_{21} + x_{31} + x_{41} \leq 1 \\
& x_{22} + x_{32} + x_{42} \leq 1 \\
& x_{23} + x_{33} + x_{43} \leq 1 \\
& x_{24} + x_{34} + x_{44} \leq 1 \\
& x_{ij} \geq 0 \ \forall \ i=2, 3, 4; j=1, 2, 3, 4.
\end{aligned}$$

The optimal solution has a value of the objective function of $\bar{z}_{F_1} = 20{,}298$, so that the effect of not having fighter F_1 available is a drop in attractiveness by 27,064 – 20,298 = 6,766. Similar values can be calculated for the other three fighters as 6,766, 3,532, and 6,766 as well as for the opponents as 6,766, 3,532, 3,532, and 6,766 for a sum of 44,426. If the event were to net $88,852, this amount could be distributed to the boxers according to their respective values. As the available dollar amount is $2 = 88,852/44,426 per unit attractiveness value, the fighters would receive $13,532, $13,532, $7,064, and $13,532 and their opponents would receive $13,532, $7,064, $7,064, and $13,532, respectively. (These values have to be understood as guarantees, winning bonuses are extra). Note that such a procedure pays the fighters and their opponents not necessarily according to their abilities. For instance, fighter F_3 receives less than any of the other fighters, even though he has the second-highest ability. The pay in this system depends on how well the individual fighters can be paired with opponents. For example, if the roster of boxers were mediocre with a single exception of a fighter whose is outstanding, then the outstanding fighter would not receive much, as his inclusion in the event will necessarily pair him with a mediocre fighter, which is not attractive, so his inclusion does not add much to the attractiveness of the event.

Three comments are in order before we discuss generalizations of the basic assignment problem. First, the most important applications of assignment problems are found in much more complex problems, in which assignment problems appear as subproblems. For instance, the famous traveling salesman problems (see, e.g., Lawler *et al*. (1985) or Gutin and Punnen (2002), Eiselt and Sandblom, 2000) is nothing but an assignment problem with a set of additional constraints that make the problem much more difficult. Some solution methods that solve these difficult problems require the repeated solution of assignment problems. Since this may have to be repeated thousands or even millions of times, a fast algorithm that solves assignment problems, is essential.

Secondly, the Hungarian Method (a solution technique first described by Kuhn (1955) and given its name as it is based on combinatorial theorems by the Hungarian mathematicians König (1931) and Egerváry (1931) has long been the

standard solution method for assignment problems. It exploits the special structure of the formulation and is highly efficient. For a description of the method, we refer to the pertinent literature; see, e.g., Eiselt *et al*. (1987) or Dantzig and Thapa (2003). Historical accounts can be found in Kuhn (1991) and Frank (2005).

A generalization of the standard assignment problem discussed above is what is known as the *generalized assignment problem* (*GAP*). It has the same structure as the standard assignment problem, but allows some parameters in the constraints to assume values other than zero or one. The generalized assignment problem can be formulated as follows:

$$\text{P: Min } z = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

$$\text{s.t. } \sum_{i=1}^{n} x_{ij} = 1 \quad \forall \quad j = 1,...,n$$

$$\sum_{j=1}^{n} a_{ij}x_{ij} = 1 \quad \forall \quad i = 1,...,n$$

$$x_{ij} = 0 \vee 1 \ \forall \ i, j = 1, \ldots, n.$$

It is apparent from the second set of constraints that the unimodularity property is lost and hence it is no longer true that all extreme points of the feasible set have integer coordinates.

The use of this problem can best be explained by use of an

Example: The public works department of a region is in the process of issuing contracts for (1) a health clinic, (2) a new sewage lagoon, (3) a local school, (4) a community center, and (5) tourist information center. Each of the three contractors who have shown interest in the projects has submitted bids for those projects he is interested in based on his abilities and costs. The matrix **C** below shows the bids that the three contractors have made on the five projects (a "–" indicates that a contractor did not bid on a project or that the agency finds the contractor unsuitable to perform this project), and the matrix **A** specifies the resource requirements of the contractors if they were to work on the projects. They may differ between contractors, based on their different uses of machines. Finally, the vector **b** shows the amount of resources available to the contractors.

$$\mathbf{A} = \begin{bmatrix} 70 & - & 40 & 30 & 50 \\ 65 & 40 & 40 & 35 & 55 \\ - & 45 & 35 & 40 & 50 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 20 & - & 10 & 9 & 15 \\ 18 & 12 & 13 & 8 & 16 \\ - & 11 & 12 & 7 & 17 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 120 \\ 100 \\ 70 \end{bmatrix}.$$

It is assumed that the government agency that awards the projects to the contractors knows **A**, **C**, and **b** and that the agency's objective is to minimize the total cost.

At optimum, the agency will award projects 3 and 5 to contractor 1, projects 1 and 4 to contractor 2, and project 2 to contractor 3. The total cost to the government (without the expected cost overruns, of course) is $\bar{z} = 62$. The contractors will then use 90, 100, and 45 units of their resources, i.e., 75%, 100%, and 64.3% of their available resources, respectively.

Finally, we should mention applications, in which the purpose is to assign items to empty positions or slots, such as companies to stores in a mall, offices in an office building, and others. The problem is that the standard assignment as it is discussed here does not allow interaction between the individual items or the positions. In most cases that is unrealistic: people walk from one store or office to another, and these movements have to be captured by the model. This can be done by what is known as the *quadratic assignment problem* (*QAP*), which is discussed in detail in, e.g., Eiselt and Sandblom (2004).

2.10 A Production – Inventory Model: A Case Study

To conclude this section, a semi-realistic production-inventory model will first be described and then formulated. Since it is a multiperiod model with two raw materials, five semi-finished goods and two finished products, quite a few variables and constraints will have to be defined. In order to keep the problem size reasonable, many simplifications have been introduced. In this manufacturing process of two types of windows we assume that the final products consist of just glass panes and wooden frames. Glue, putty, etc., will not be considered, and manpower requirements are also neglected.

The formulation of this model will shed some light on the intricacies and difficulties encountered in modeling of real world problems. First consider the main aspects of the problem.

(1) General Structure:

Glass sheets of size 10' × 10' and wooden boards of length 20' are purchased and stored in a warehouse (INV 1). The glass is processed on machine M1a where each 10' × 10' sheet is cut into either four 4' × 5' sheets or nine 3' × 3' sheets; while the wood is processed on machine M1b, which cuts each 20' piece into six 3' lengths, five 4' lengths, or four 5' lengths. The resulting semi-finished products are stored in the second warehouse (INV 2). From there, the materials are taken to machine M2, on which the two finished products, i.e. the 5' × 4' and 3' × 3' windows are assembled by using the glass and wood available. These two types of windows are stored in the third warehouse (INV 3) and from there they are sold.

The planning horizon is two periods.

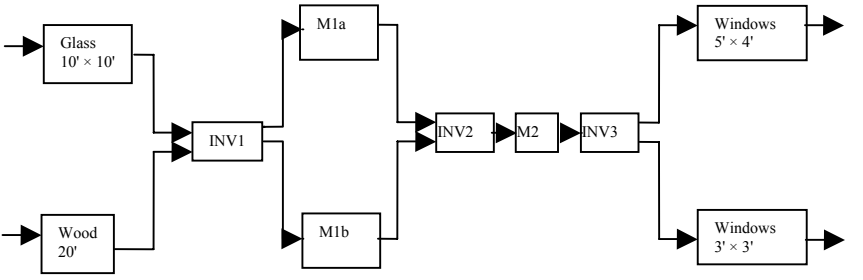


Figure 2.7

(2) The Materials:

Table 2.9 shows the availability of the two raw materials, their expected prices, and their respective weights.

Table 2.9

Item	Availability (both periods) in units		Expected price per unit		Weight
	Lower Bound	Upper Bound	Period 1	Period 2	
Glass 10' × 10' (× $\frac{1}{50}$ ')	100	500	\$20	\$22	4 ounces per ft^2
Wood 20' × $\frac{1}{4}$ ' × $\frac{1}{20}$ ')	0	2000	\$6	\$7	3 ounces per ft^2

The volume of the materials can be calculated from the above parameters.

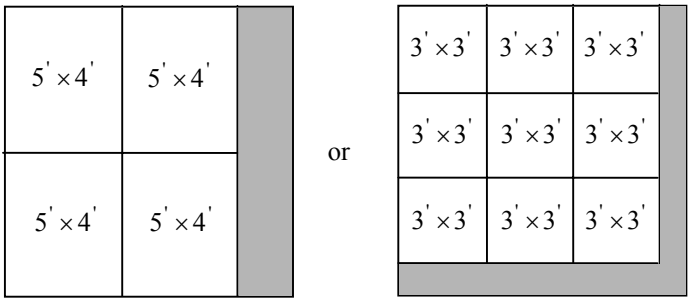
(3) The Machines:

As shown in figure 2.7, there are two types of machines: while M1a and M1b cut the raw materials to their appropriate sizes, M2 assembles the semi-finished products to the final products. The operating costs, processing times, and capacities of the machines are shown in Table 2.10. The data in the table are valid for both periods. There are no losses (damages) in the production process.

Table 2.10

	Operating costs	Processing times	Capacities
M1a	\$100 per hour	1 cut (any length) in 20 seconds	$13\frac{20}{60}$ hours per period
M1b	\$50 per hour	1 cut (any length) in 8 seconds	15 hours per period
M2	\$4.50 per window	1 minute per window	15 hours per period

M1a: The glass-cutting machine



4 cuts are necessary 10 cuts are necessary

Figure 2.8

The reason for the number of cuts required to produce the above patterns is the assumption that we can only cut one sheet at a time.

M1b:

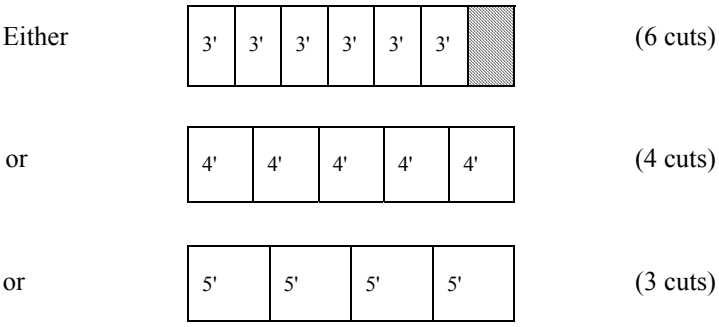


Figure 2.9

(4) The Warehouses:

Table 2.11

	Capacities		Losses (when taken out)	Initial inventory (at the beginning of period 1)	Final inventory (at the end of period 2)
	Weight oz.	Volume ft^3			
INV1	40,000	200	5% of glass (complete loss)	50 sheets of 10' × 10' glass	40 sheets of 10' × 10' glass 20 boards of 20' wood
INV2	10,000	300	none	none	20 sheets of 3' × 3' glass 10 boards of 3' wood 10 boards of 4 ' wood 10 boards of 5' wood
INV 3	20,000	100	3% of windows (complete loss)	20 units of 5' × 4' windows	50 units of 5' × 4' windows 30 units of 3'× 3' windows

Inventory costs (between periods 1 and 2 only): 1% of the value of the raw material, semi-finished, or finished product stored in period 1. No costs are incurred for items in stock at the end of period 2.

(5) The Finished Products:

The demands and unit prices of the two types of windows in the two periods are shown in Table 2.12.

Table 2.12

Products	Demand (Upper bound) in units		Unit prices ($)	
	Period 1	Period 2	Period 1	Period 2
Windows 5' × 4'	1500	2000	75	80
Windows 3' × 3'	2000	2000	45	51

(6) The Objective:

Maximize total profit and report an optimal solution to management.

In order to formulate the above problem we will examine the input of materials into warehouses and machines, the output of materials of warehouses and machines, the levels of inventories, and the use of the machines separately. The formulation offered below follows the flow of materials through the system. As everywhere in practice, a mnemonic code for the variables will be used. In each of the following eight parts, the variables will be defined, the constraints will be set up and the corresponding costs or revenues (denoted by "profit contribution" and later to be put together in the objective function) will be formulated. The nonnegativity of all variables is assumed but not explicitly stated.

Part 1: The Supplies

Define *IG1* and *IG2* as the input of glass into the system in period 1 and 2, respectively, and let *IW1* and *IW2* be the corresponding variables for the input of wood in the two periods. The constraints for glass and wood are then formulated as:

$$IG1 \geq 100 \quad (1)$$
$$IG1 \leq 500 \quad (2)$$
$$IG2 \geq 100 \quad (3)$$
$$IG2 \leq 500 \quad (4)$$

for glass and

$$IW1 \leq 2000 \quad (5)$$
$$IW2 \leq 2000 \quad (6)$$

for wood.

Since all purchased units are put into inventory 1 right away, we can define *IGI11* and *IGI12* as the input of glass into inventory 1 in period 1 and 2, respectively, and *IWI11* and *IWI12* as the input of wood in the two periods. Hence

$$IGI = IGI11 \quad (7)$$
$$IW1 = IWI11 \quad (8)$$

and

$$IG2 = IGI12 \quad (9)$$
$$IW2 = IWI12 \quad (10)$$

The total costs for the supplies in both periods are

$$(20)IG1 + (22)IG2 + (6)IW1 + (7)IW2 \qquad \text{(Profit contribution)}$$

Part 2: Inventory 1

Let $OGI11$ ($OGI12$) and $OWI11$ ($OWI12$) denote the output of glass and wood of inventory 1 in the first (second) period respectively. Define $LI1G12$ and $LI1W12$ as the level of inventory 1 concerning glass between period 1 and 2 (the number of 10' × 10' sheets of glass in INV1 between the periods) and the inventory 1 level concerning wood (the number of 20' boards of wood) between the periods and let $LI1G23$ and $LI1W23$ be the respective inventories—between the second and third period, i.e., the end of the planning horizon—of glass and wood. Note that $LI1G12$ and $LI1W12$ can be expressed as the beginning inventory plus the input into INV1 in period 1 minus the output of INV1 in period 1, and hence we can write

$$LI1G12 = 50 + IGI11 - OGI11 \tag{11}$$
$$LI1W12 = 0 + IWI11 - OWI11 \tag{12}$$

In the second period, the beginning inventories are $LI1G12$ and $LI1W12$ respectively, so that the inventory levels at the end of the second period (or, equivalently, between the second and third period) are

$$LI1G23 = LI1G12 + IGI12 - OGI12 \tag{13}$$
$$LI1W23 = LI1W12 + IWI12 - OWI12 \tag{14}$$

Moreover, the requirements for the final inventories can be written as

$$LI1G23 \geq 40 \tag{15}$$
$$LI1W23 \geq 20 \tag{16}$$

In order to establish the capacity constraints for the warehouses, we will consider the weight constraints first and the volume constraints later. Considering the fact that 1ft^2 of glass weighs 4 ounces, each 10' × 10' sheet of glass has a weight of 400 ounces and since 1 ft of wood weighs 3 ounces, each 20' board of wood has a weight of 60 ounces. Using the weight constraint of 40,000 ounces of INV1, we can write

$$(400)LI1G12 + (60)LI1W12 \leq 40{,}000 \tag{17}$$

for period 1 and

$$(400)LI1G23 + (60)LI1W23 \leq 40{,}000 \tag{18}$$

for period 2.

Since each sheet of glass has dimensions 10' × 10' × $\frac{1}{50}$', it has a volume of 2 ft^3; each board of 20' × $\frac{1}{4}$' × $\frac{1}{20}$' wood has a volume of $\frac{1}{4}$ ft^3, and therefore using the allowed volume of 200 ft^3 for INV1, we state

$$(2)LI1G12 + \tfrac{1}{4} LI1W12 \leq 200 \tag{19}$$

for period 1, and

$$(2)LI1G23 + \tfrac{1}{4} LI1W23 \leq 200 \tag{20}$$

for period 2.

Finally, since the inventory costs are 1% of the price of the stored material and the prices were \$20 and \$6 in period 1, the costs for INV1 are

$$\tfrac{20}{100} LI1G12 + \tfrac{6}{100} LI1W12. \qquad \text{(Profit contribution)}$$

Part 3: Machines 1a and 1b: The Transformations

Since 5% of the glass taken out of the inventory is lost (e.g. broken), we have to distinguish between the output of INV1 and the input into machine M1a. Defining *IM1a1* (*IM1a2*) and *IM1b1* (*IM1b2*) as the input of glass and wood into machines M1a and M1b, respectively, in the first (second) period, we obtain

$$IM1a1 = \tfrac{95}{100} OGI11 \tag{21}$$

(the constraint that governs the loss of glass in period 1), and

$$IM1a2 = \tfrac{95}{100} OGI12 \tag{22}$$

(the constraint the models the loss of glass in period 2).

Since there is no loss of wood, we have

$$IM1b1 = OWI11 \tag{23}$$

and

$$IM1b2 = OWI12. \tag{24}$$

In order to discuss the qualitative transformations in the cutting operations on machines M1a and M1b, we define *OM1a541* (*OM1a331*) and *OM1a331* (*OM1a332*) as the output of 5' × 4' and 3' × 3' glass sheets of machine M1a in period 1 (period 2) respectively. For machine M1b, we define *OM1b31*

(*OM1b32*), *OM1b41*, (*OM1b42*), and *OM1b51* (*OMb152*) as the output of 3ft, 4 ft and 5ft boards of machine M1b in period 1 (period 2) respectively. Since each transformation of a 10' × 10' sheet of glass results in either four 5' × 4' sheets or in nine 3' × 3'sheets, we can write

$$\tfrac{1}{4}\,OM1a541 + \tfrac{1}{9}\,OM1a331 = IM1a1 \tag{25}$$

for period 1, and

$$\tfrac{1}{4}\,OM1a542 + \tfrac{1}{9}\,OM1a332 = IM1a2 \tag{26}$$

for period 2.

Consider now machine M1b on which the wood is cut; each 20 ft board is either cut into six 3 ft pieces, five 4 ft pieces, or four 5 ft pieces. Hence

$$\tfrac{1}{6}\,OM1b31 + \tfrac{1}{5}\,OM1b41 + \tfrac{1}{4}\,OM1b51 = IM1b1 \tag{27}$$

for period 1, and

$$\tfrac{1}{6}\,OM1b32 + \tfrac{1}{5}\,OM1b42 + \tfrac{1}{4}\,OM1b52 = IM1b2 \tag{28}$$

for period 2.

Since between machines M1a and M1b and inventory 2 there are assumed to be no losses, the output of M1a and M1b equals the input into INV2. Let *IG54I21* (*IG54I22*), *IG33I21* (*IG33I22*), *IW3I21* (*IW3I22*), *IW4I21* (*IW4I22*) and *IW5I21* (*IW5I22*) denote the input into INV2 of glass 5' × 4', glass 3' × 3', wood 3', wood 4', and wood 5' in period 1 (period 2) respectively. The nonexistence of losses of the two sizes of glass is then written as

$$IG54I21 = OM1a541 \tag{29}$$
$$IG33I21 = OM1a331 \tag{30}$$

for period 1 and

$$IG54I22 = OM1a542 \tag{31}$$
$$IG33I22 = OM1a332 \tag{32}$$

for period 2, respectively. For the three lengths of boards, we obtain the conditions

$$IW3I21 = OM1b31 \tag{33}$$
$$IW4I21 = OM1b41 \tag{34}$$
$$IW5I21 = OM1b51 \tag{35}$$

for period 1, and

$$IW3I22 = OM1b32 \tag{36}$$
$$IW4I22 = OM1b42 \tag{37}$$
$$IW5I22 = OM1b52 \tag{38}$$

for period 2, respectively.

Part 4: Machines 1a and 1b: The Utilization

For this part, the only new variables needed are *HM1a1* (*HM1a2*) and *HM1b1* (*HM1b2*) which denote the number of hours of used time on the machines M1a and M1b in period 1 (period 2), respectively. Consider machine M1a first. Note that *HM1a1* and *HM1a2* have to be expressed in terms of the output and not the input of machine M1a, since from the input one cannot tell how the glass is cut and how many cuts are needed. Since for each 10' × 10' sheet of glass which is cut into four 5' × 4' sheets, exactly four cuts are required, the number of cuts for this transformation equals *OM1a541* (and *OM1a542* for period 2). Each cut requires $\frac{1}{3}$ of a minute, so that the time needed to transform 10' × 10' glass into 5' × 4' sheets equals $\frac{1}{3}\,OM1a541$ for each period 1, and similarly for period 2. On the other hand, for the transformation of each 10' × 10' sheet of glass into nine 3' × 3' sheets, 10 cuts are needed so that the total number of cuts for this type of transformation is $\frac{10}{9}\,OM1a331$ (or $\frac{10}{9}\,OM1a332$ for period 2). Again, $\frac{1}{3}$ of a minute is required per cut so that the total time for the transformation of a 10' × 10' sheet of glass into 3' × 3' glass is $\frac{10}{9}\,\frac{1}{3}\,OM1a331$ for period 1, and similarly for period 2. Hence we can write

$$HM1a1 = \tfrac{1}{3}\,OM1a541 + \tfrac{10}{27}\,OM1a331 \tag{39}$$

for period 1, and

$$HM1a2 = \tfrac{1}{3}\,OM1a542 + \tfrac{10}{27}\,OM1a332 \tag{40}$$

for period 2.

Since the capacity of machine M1a is $13\frac{20}{60}$ hours or 800 minutes in both periods, we obtain

$$HM1a1 \leq 800 \tag{41}$$

$$HM1a2 \leq 800. \tag{42}$$

Similar considerations have to be made for machine M1b. Consider the transformation of a 20ft board into six 3 ft boards. Since exactly six cuts are needed, the number of cuts for this transformation equals the number of 3 ft boards produced, i.e., *OM1b31* for period 1 and *OM1b32* for period 2. Since $\frac{8}{60}$ minutes are needed per cut, the total time required to cut 20 ft boards into 3 ft boards is $\frac{8}{60}$ *OM1b31* for period 1, and similarly for period 2.

For the transformation of a 20 ft board into five 4 ft boards, four cuts are needed, hence the total number of cuts required for this type of transformation is $\frac{4}{5}$ *OM1b41* (and $\frac{4}{5}$ *OM1b42* for period 2). Each cut takes $\frac{8}{60}$ minutes, so the total time required for the transformation of 20 ft boards into 4 ft boards is $\frac{8}{60}\ \frac{4}{5}$ *OM1b41* for period 1 and $\frac{8}{60}\ \frac{4}{5}$ *OM1b42* for period 2. Finally, we need three cuts to transform one 20 ft board into four 5 ft boards, and hence a total of $\frac{3}{4}$ *OM1b51* (and $\frac{3}{4}$ *OM1b52* for period 2) cuts are needed, so that the total time required for making this kind of transformation is $\frac{8}{60}\ \frac{3}{4}$ *OM1b51* for period 1 and $\frac{8}{60}\ \frac{3}{4}$ *OM1b52* for period 2. Consequently, the utilization time of machine M1b is

$$HM1b1 = \frac{8}{60}\,OM1b31 + \frac{8}{60}\,\frac{4}{5}\,OM1b41 + \frac{8}{60}\,\frac{3}{4}\,OM1b51 \qquad (43)$$

for period 1, and

$$HM1b2 = \frac{8}{60}\,OM1b32 + \frac{8}{60}\,\frac{4}{5}\,OM1b42 + \frac{8}{60}\,\frac{3}{4}\,OM1b52 \qquad (44)$$

for period 2.

Since machine M1b has a capacity of 15 hours or 900 minutes in each period, we can formulate

$$HM1b1 \leq 900 \qquad (45)$$
$$HM1b2 \leq 900 \qquad (46)$$

The machine operating costs are

$$\frac{100}{60}\,HM1a1 + \frac{100}{60}\,HM1b2 \qquad \text{(Profit contribution)}$$

for machine M1a, and

$$\frac{50}{60}\,HM1b1 + \frac{50}{60}\,HM1b2. \qquad \text{(Profit contribution)}$$

for machine M1b.

Part 5: Inventory 2

Let *OG54I21* (*OG54I22*) be the output of 5' × 4' glass out of inventory 2 in period 1 (period 2), let *OG33I21* (*OG33I22*) be the output of 3' × 3' glass out of inventory 2 in period 1, and let *OW3I21* (*OW3I22*), *OW4I21* (*OW4I22*) and *OW5I21* (*OW5I22*) be the output of 3 ft, 4 ft and 5 ft wood out of INV2 in period 1 (period 2), respectively. Moreover, let *LI2G5412* (*LI2G5423*), *LI2G3312*, (*LI2G3323*), *LI2W312*, (*LI2W323*), *LI2W412* (*LI2W423*) and *LI2W512* (*LI2W523*) denote the level of inventory 2 of 5' × 4' glass, 3' × 3' glass, 3 ft wood, 4 ft wood, and 5 ft wood between periods 1 and 2 (between periods 2 and 3 or at the end of period 2). Since no beginning inventory is given, we can state that the level of INV2 between periods 1 and 2 equals the input into INV2 in period 1 minus the output out of INV2 in period 1 for all semi-finished products, i.e.

$$LI2G5412 = IG54I21 - OG54I21 \quad (47)$$
$$LI2G3312 = IG33I21 - OG33I21 \quad (48)$$
$$LI2W312 = IW3I21 - OW3I21 \quad (49)$$
$$LI2W412 = IW4I21 - OW4I21 \quad (50)$$
$$LI2W512 = IW5I21 - OW5I21 \quad (51)$$

The level of inventory 2 at the end of period 2 equals the beginning inventory in period 2—which is *LI2G5412*, *LI2G3312*, etc.—plus the input into INV2 in period 2 minus the output out of INV2 in period 2, i.e.,

$$LI2G5423 = LI2G5412 + IG54I22 - OG54I22 \quad (52)$$
$$LI2G3323 = LI2G3312 + IG33I22 - OG33I22 \quad (53)$$
$$LI2W323 = LI2W312 + IW3I22 - OW3I22 \quad (54)$$
$$LI2W423 = LI2W412 + IW4I22 - OW4I22 \quad (55)$$
$$LI2W523 = LI2W512 + IW5I22 - OW5I22 \quad (56)$$

In order to satisfy the requirements for the final inventory, we must have

$$LI2G3323 \geq 20 \quad (57)$$
$$LI2W323 \geq 10 \quad (58)$$
$$LI2W423 \geq 10 \quad (59)$$
$$LI2W523 \geq 10 \quad (60)$$

Below, we formulate the capacity constraints related to INV2. For that purpose, we first express the total weight of all items in INV2. Each 5' × 4' sheet of glass covers an area of 20 ft^2, each ft^2 of glass has a weight of four ounces, and hence each of these 5' × 4' sheets has a weight of 80 ounces; accordingly, each 3' × 3' sheet of glass (i.e., 9 ft^2) weighs (9)(4) = 36 ounces. Similarly, since each linear foot of wood weighs 3 ounces, each 3, 4 and 5 ft board has a weight of 9, 12 and 15 ounces, respectively. Considering the 10,000 ounce capacity of INV2, we can state that

$$(80)LI2G5412 + (36)LI2G3312 + (9)LI2W312 + (12)LI2W412 + (15)LI2W512 \leq 10{,}000 \quad (61)$$

for period 1, and

$$(80)LI2G5423 + (36)LI2G3323 + (9)LI12W323 + (12)LI2W423 + (15)LI2W523 \leq 10{,}000 \quad (62)$$

for period 2.

As far as volume is concerned, each 5' × 4' sheet of glass has a volume of $(5)(4)\frac{1}{50} = \frac{2}{5}$ ft^3; the volume of each 3' × 3' sheet of glass is $(3)(3)\frac{1}{50} = \frac{9}{50}$ ft^3. The corresponding volumes of the 3 ft, 4 ft and 5 ft wood are $(3)\frac{1}{4}\frac{1}{20} = \frac{3}{80}$ ft^3, $(4)\frac{1}{4}\frac{1}{20} = \frac{1}{20}$ ft^3 and $(5)\frac{1}{4}\frac{1}{20} = \frac{1}{16}$ ft^3 respectively. Considering the capacity of INV2 of 300 ft^3, we state

$$\frac{2}{5}LI2G5412 + \frac{9}{50}LI2G3312 + \frac{3}{80}LI2W312 + \frac{1}{20}LI2W412 + \frac{1}{16}LI2W512 \leq 300 \quad (63)$$

for period 1 and

$$\frac{2}{5}LI2G5423 + \frac{9}{50}LI2G3323 + \frac{3}{80}LI2W323 + \frac{1}{20}LI2W423 + \frac{1}{16}LI2W523 \leq 300 \quad (64)$$

for period 2.

Finally, we establish the costs for INV2, based on the inventory level between period 1 and 2, as well as the costs of materials in period 1. Since one 10' × 10' sheet of glass has a price of \$20 in period 1 and four 5' × 4' sheets were made out of each, the cost of a 5' × 4' sheet is \$5, and that of a 3' × 3' sheet is $\$\frac{20}{9}$. Note that the cost for the waste is included in these pieces. Since one 20 ft board of wood costs \$6 in period 1, one 3 ft board costs \$1, one 4 ft board costs $\$\frac{6}{5}$ and one 5 ft board costs $\$\frac{6}{4}$. In addition to the costs of the materials, the proportional production costs for the various types of materials on M1a and M1b have to be considered. First M1a will be discussed. Since one cut takes 20 seconds and one machine hour costs \$100, we have to consider $\$\frac{5}{9}$ per cut.

- Cutting 10' × 10' sheets of glass into 5' × 4' sheets: Result: 4 sheets (of 5' × 4'), using 4 cuts. Hence $\frac{5}{9}$ [\$/sheet] are the proportional machine costs.

- Cutting 10' × 10' sheets of glass into 3' × 3' sheets: Result: 9 sheets of size 3' × 3', using 10 cuts. Hence $\frac{10}{9}\ \frac{5}{9} = \frac{50}{81}$ [$/sheet] are the proportional machine costs.

- Machine M1b: Each cut takes 8 seconds and a machine-hour costs $50, hence $\frac{1}{9}$ [$/cut] has to be considered.

- Cutting 20 ft boards of wood into 3 ft boards: Result: 6 boards of 3 ft wood, requiring 6 cuts. Hence $\frac{1}{9}$ [$/board] are the proportional machine costs.

- Cutting 20 ft boards of wood into 4 foot boards: Result: 5 boards of length 4 ft), requiring 4 cuts. Hence $\frac{4}{5}\ \frac{1}{9} = \frac{4}{45}$ [$/board] are the proportional machine costs.

- Cutting 20 ft boards of wood into 5 ft boards: Result: 4 boards of 5 ft wood, requiring 3 cuts. Hence $\frac{3}{4}\ \frac{1}{9} = \frac{1}{12}$ [$/board] are the proportional machine costs.

Hence the values of the semi-finished products (including purchase price and production costs) are:

- For 5' × 4' glass: $\$\,5\frac{5}{9}$
- for 3' × 3' glass: $\$\,2\frac{68}{81}$
- for 3 ft wood: $\$\,1\frac{1}{9}$
- for 4 ft wood: $\$\,1\frac{13}{45}$, and
- for 5 ft wood: $\$\,1\frac{7}{12}$

One percent of these costs is incurred for each item in store between period 1 and 2. We then obtain the costs for INV2 as follows:

$$\frac{50}{9}\frac{1}{100}LI2G5412 + \frac{230}{81}\frac{1}{100}LI2G3312 + \frac{10}{9}\frac{1}{100}LI2W312 + \frac{58}{45}\frac{1}{100}LI2W412 + \frac{19}{12}\frac{1}{100}LI2W512 \qquad \text{(Profit contribution)}$$

Part 6: Machine 2

Let *IG54M21* (*IG54M22*), *IG33M21* (*IG33M22*), *IW3M21* (*IW3M22*), *IW4M21* (*IW4M22*) and *IW5M21* (*IW5M22*) be defined as the input of 5' × 4' glass, 3' × 3' glass, 3 ft, 4 ft and 5 ft wood into machine M2 in period 1 (period 2), respectively. Since there are no losses between INV2 and M2, the following equalities hold:

$$IG54M21 = OG54I21 \quad (65)$$
$$IG33M21 = OG33I21 \quad (66)$$
$$IW3M21 = OW3I21 \quad (67)$$
$$IW4M21 = OW4I21 \quad (68)$$
$$IW5M21 = OW5I21 \quad (69)$$

for period 1, and

$$IG54M22 = OG54I22 \quad (70)$$
$$IG33M22 = OG33I22 \quad (71)$$
$$IG33M22 = OG33I22 \quad (72)$$
$$IW3M22 = OW3I22 \quad (73)$$
$$IW4M22 = OW4I22 \quad (74)$$
$$IW5M22 = OW5I22 \quad (75)$$

In order to describe the output of machine M2, we define *OWI54M21* (*OWI54M22*) and *OWI33M21* (*OWI33M22*) as the output of 5' × 4' and 3' × 3' windows out of machine M2 in period 1 (period 2) respectively. Since there are no losses during the assembly of the windows on machine M2, the number of 5' × 4' (3' × 3') windows leaving M2 equals the number of 5' × 4' (3' × 3') units of glass, sent into M2, i.e.,

$$OWI54M21 = IG54M21 \text{ and} \quad (76)$$
$$OWI33M21 = IG33M21 \quad (77)$$

for period 1, and

$$OWI54M22 = IG54M22 \text{ and} \quad (78)$$
$$OWI33M22 = IG33M22 \quad (79)$$

for period 2.

On the other hand, since each 5' × 4' window requires—in addition to one 5' × 4' sheet of glass—exactly two boards of 4 ft wood and two boards of 5 ft wood, we can write

$$(2)OWI54M21 = IW4M21 \text{ and} \quad (80)$$
$$(2)OWI54M21 = IW5M21 \quad (81)$$

for period 1, and

$$(2)OWI54M22 = IW4M22 \text{ and} \quad (82)$$
$$(2)OWI54M22 = IW5M22 \quad (83)$$

for period 2.

For each 3' × 3' window we need—in addition to one 3' × 3' sheet of glass—four boards of 3 ft wood, and so we can state:

$$(4)OWI33M21 = IW3M21 \tag{84}$$

for period 1, and

$$(4)OWI33M22 = IW3M22 \tag{85}$$

for period 2.

Moreover, since one minute is required for the assembly of each type of window on M2, the total time used for the assembly of windows (in minutes) equals the number of windows assembled in the corresponding period. Defining *HM21* and *HM22* as the utilization of machine M2 in period 1 and 2 respectively, we can write

$$HM21 = OWI54M21 + OWI33M21 \tag{86}$$

for period 1, and

$$HM22 = OWI54M22 + OWI33M22 \tag{87}$$

for period 2.

The capacity of M2, *viz*., 15 hours, can be incorporated in the problem as

$$HM21 \leq 900 \tag{88}$$
$$HM22 \leq 900 \tag{89}$$

The costs for the operation of machine M2 are easily formulated since they are expressed in terms of "$ per window"; hence they are

$$4½\ OWI54M21 + 4½\ OWI54M22 + 4½\ OWI33M21 + 4½\ OWI33M22 \quad \text{(Profit contribution)}$$

Part 7: Inventory 3

Define *IWI54I31* (*IWI54I32*) and *IWI33I31* (*IWI33I32*) as the input of 5' × 4' and 3' × 3' windows into INV3 in period 1 (period 2) respectively. Since no losses occur between M2 and INV3, we can state

$$IWI54I31 = OWI54M21 \tag{90}$$
$$IWI33I31 = OWI33M21 \tag{91}$$

in period 1, and

$$IWI54I32 = OWI54M22 \quad (92)$$
$$IWI33I32 = OWI33M22 \quad (93)$$

in period 2, respectively.

Let *OWI54I31* (*OWI54I32*) and *OWI33I31* (*OWI33I32*)denote the output of 5' × 4' and 3' × 3' windows of inventory INV3 in period 1 (period 2) respectively; furthermore, define *LI3WI5412* (*LI3WI5423*) and *LI3WI3312* (*LI3WI3323*) as the level of inventory 3 of 5' × 4' and 3' × 3' windows between periods 1 and 2 (between periods 2 and 3) respectively. Considering the beginning inventory of twenty 5' × 4' and no 3' × 3' windows, the inventory levels between periods 1 and 2 are

$$LI3W5412 = 20 + IWI54I31 - OWI54I31, \text{ and} \quad (94)$$

and

$$LI3WI3312 = 0 + IWI33I31 - OWI33I31. \quad (95)$$

Considering the fact that *LI3WI5412* and *LI3WI3312* are the beginning inventories for period 2, we can write the ending inventories of period 2 as

$$LI3WI5423 = LI3WI5412 + IWI54I32 - OWI54I32, \text{ and} \quad (96)$$
$$LI3WI3323 = LI3WI3312 + IWI33I32 - OWI33I32. \quad (97)$$

Since the required final inventory of 5' × 4' and 3' × 3' windows are 50 and 30 units respectively, we obtain

$$LI3WI5423 \geq 50 \text{ and} \quad (98)$$
$$LI3WI3323 \geq 30. \quad (99)$$

Now the constraints for the capacity of INV3 will be established. First the weights of the types of windows will be discussed. Each 5' × 4' window consists of a 5' × 4' sheet of glass—weight 80 ounces—as well as two boards of 5 ft wood and two boards of 4 ft wood, i.e. a total of 18 ft wood which weighs (18)(3) = 54 ounces. Hence each 5' × 4' window has a weight of 134 ounces. Each 3' × 3' window consists of a 3' × 3' sheet of glass—weight 36 ounces—and four boards of 3 ft wood, totaling 12 ft of wood which has a weight of 36 ounces: hence each 3' × 3' window weighs 72 ounces. Since INV3 has a capacity of 20,000 ounces, we can state

$$(134)LI3WI5412 + (72)LI3WI3312 \leq 20{,}000 \quad (100)$$

for period 1 and

$$(134)LI3WI5423 + (72)LI3WI3323 \leq 20{,}000 \qquad (101)$$

for period 2.

The volume of the two types of windows can be calculated as follows. Each 5' × 4' window includes a 5' × 4' × $\frac{1}{50}$' sheet of glass which has a volume of $\frac{2}{5}$ ft^3 as well as (2)(5) + (2)(4) = 18ft of wood, i.e., 18' × 1/4' × $\frac{1}{20}$', hence a volume of $\frac{9}{40}$ ft^3, so the total volume is $\frac{5}{8}$ ft^3. Similarly, each 3' × 3' window consists of 3' × 3' × $\frac{1}{50}$' = $\frac{9}{50}$ ft^3 of glass and 12' × 1/4' × $\frac{1}{20}$' = $\frac{3}{20}$ ft^3 of wood, so the total volume is $\frac{33}{100}$ ft^3. Since INV3 has a capacity of 100 ft ^{3}, the volume constraints are

$$\tfrac{5}{8}\,LI3WI5412 + \tfrac{33}{100}\,LI3WI3312 \leq 100 \qquad (102)$$

for period 1, and

$$\tfrac{5}{8}\,LI3WI5423 + \tfrac{33}{100}\,LI3WI3323 \leq 100 \qquad (103)$$

for period 2.

Here the volume of a window is regarded to be equal to the sum of the volumes of its parts, although this is a simplification which might not be true in practice.

The costs for the items in INV3 are based on the purchase price and the production costs of the finished products. Using the costs of the2 semi-finished products (see part 5), we obtain the following values:

- 5' × 4' windows: one 5' × 4' sheet of glass: $\$5\frac{5}{9}$
  - 2 boards of 4 ft wood: $(2)\$1\frac{13}{45}$
  - 2 boards of 5 ft wood: $(2)\$1\frac{7}{12}$
  - Assembly of one window: $\$4\frac{1}{2}$
  - Total costs: $\$15\frac{4}{5}$

- 3' × 3' window: one 3' × 3' sheet of glass: $\$2\frac{68}{81}$
  - 4 boards of 3 ft wood $(4)\$1\frac{1}{9}$
  - Assembly of 1 window: $\$4\frac{1}{2}$
  - Total costs: $\$11\frac{127}{162}$

Hence the inventory costs in INV3 are

$$\frac{79}{5}\frac{1}{100}LI3WI5412 + \frac{1909}{162}\frac{1}{100}LI3WI3312 \qquad \text{(Profit contribution)}$$

Part 8: The Sales

Define *SWI541* (*SWI542*) and *SWI331* (*SWI332*) as the sales of 5' × 4' and 3' × 3' windows in period 1 (period 2) respectively. The fact that 3% of the windows are lost if they leave INV3 is expressed as

$$SWI541 = \frac{97}{100}OWI54I31 \text{ and} \tag{104}$$
$$SWI331 = \frac{97}{100}OWI33I31 \tag{105}$$

for period 1, and

$$SWI542 = \frac{97}{100}OWI54I32 \text{ and} \tag{106}$$
$$SWI332 = \frac{97}{100}OWI33I32 \tag{107}$$

for period 2.

Moreover, the upper bounds for the demand have to be respected, i.e.,

$$SWI541 \leq 1{,}500 \tag{108}$$
$$SWI331 \leq 2{,}000 \tag{109}$$
$$SWI542 \leq 2{,}000 \tag{110}$$
$$SWI332 \leq 2{,}000 \tag{111}$$

Finally, the revenue from these sales is

$$(75)SWI541 + (45)WI331 + (80)WI542 + (51)WI332$$

(Profit contribution)

The above problem has a total of 62 variables and just over 100 structural constraints, making it a small problem by today's standard, at least from a computational point of view. The solution is displayed in the following figures, showing the material flows in the two periods. Note that the value of the objective function—the total profit—is \$101,909.27 for both periods combined.

The solution is displayed in Figures 2.10 – 2.12.

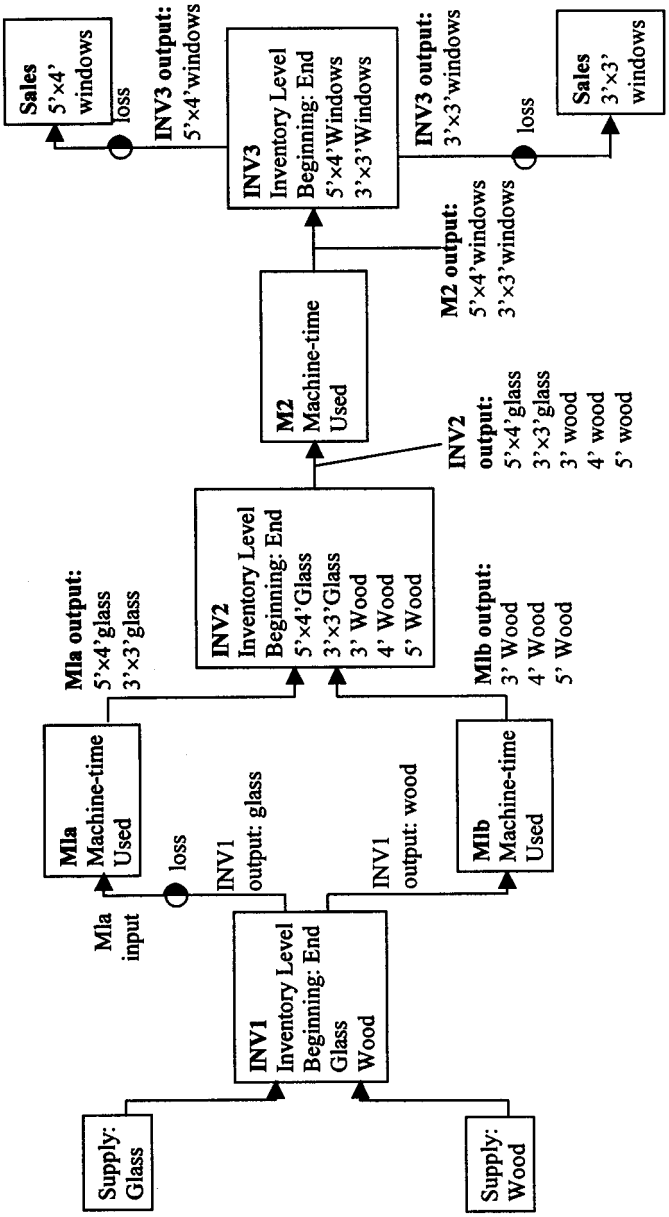

Supply: Glass
Supply: Wood
INV1 Inventory Level Beginning: End Glass Wood
Mla input
loss
INV1 output: glass
INV1 output: wood
Mla Machine-time Used
Mlb Machine-time Used
Mla output: 5'×4'glass 3'×3'glass
Mlb output: 3' Wood 4' Wood 5' Wood
INV2 Inventory Level Beginning: End 5'×4'Glass 3'×3'Glass 3' Wood 4' Wood 5' Wood
M2 Machine-time Used
INV2 output: 5'×4'glass 3'×3'glass 3' wood 4' wood 5' wood
M2 output: 5'×4'windows 3'×3'windows
INV3 Inventory Level Beginning: End 5'×4'Windows 3'×3'Windows
INV3 output: 5'×4'windows
INV3 output: 3'×3'windows
loss
loss
Sales 5'×4' windows
Sales 3'×3' windows

Figure 2.10: Legend

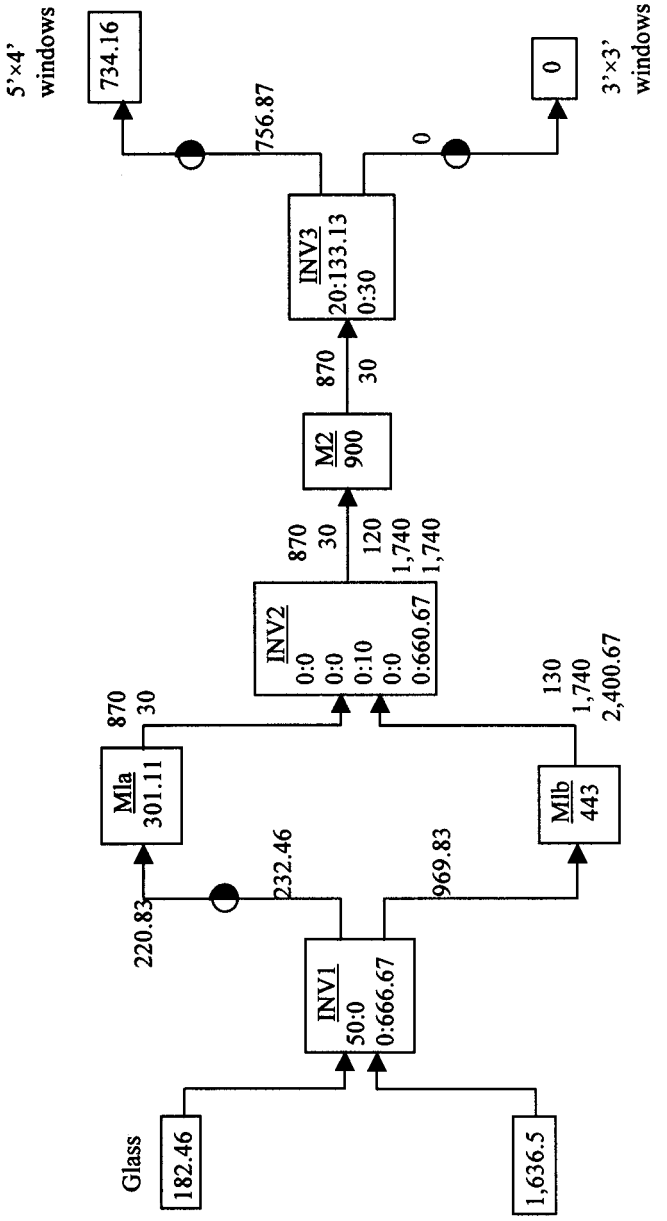


Figure 2.11: Period 1

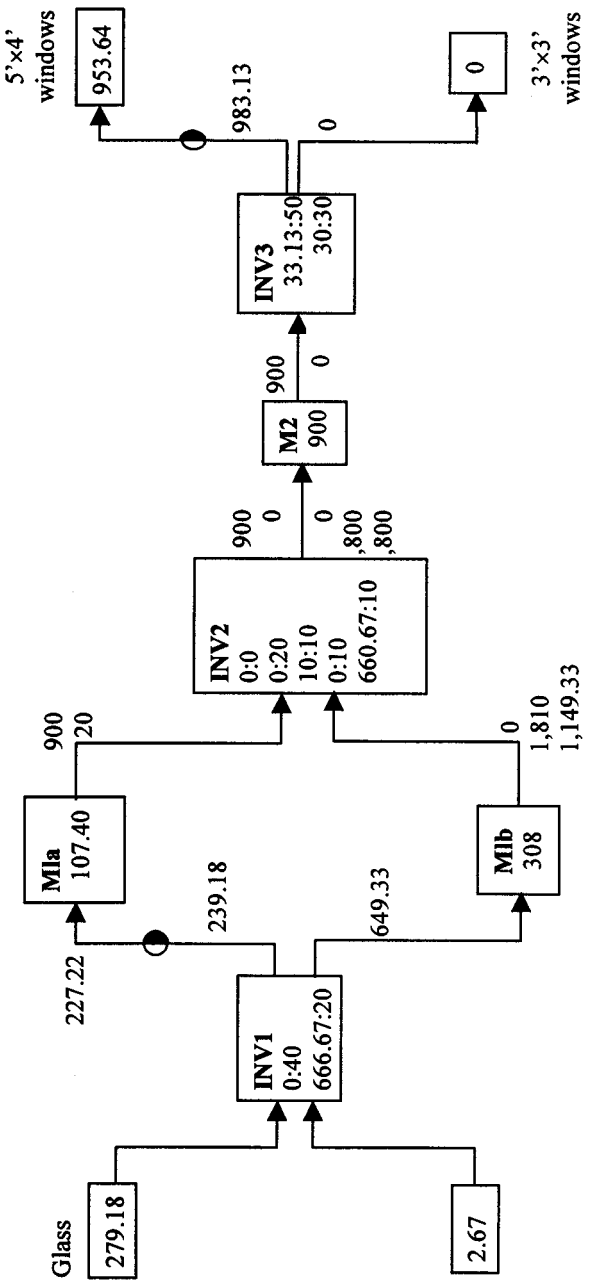


Figure 2.12: Period 2

Without discussing the solution in greater detail, some comments should be made:

(a) All three warehouses are filled up to capacity (concerning weight) between periods 1 and 2.

(b) Machine M2 works at capacity in both periods.

(c) 3' × 3' windows are not produced for sale in any of the two periods.

Note that observations (a) and (b) indicate bottlenecks in the production-inventory model; in future planning one may consider increasing the pertinent capacities, given that expansion is possible and the other parameters remain unchanged. Moreover, as the above observation (c) indicates that the price ratio for 5' × 4' and 3' × 3' windows is in favor of the 5' × 4' type, a price increase for 3' × 3' windows may be considered, provided the market is willing to accept it.

One more general note seems to be in order here. Usually, there is more than one way to formulate a given problem and one might try to find an "optimal" formulation. In order to find this "optimum," the objective has to be stated first. Basically, two objectives can be thought of: One is to make the model as small as possible by including the least number of constraints and variables; another one is to formulate the problem so that the solution provides the best documentation possible. Unfortunately, a good documentation will almost always require a larger number of variables, so that both objectives are not likely to be optimized with a single formulation. For instance, the variables *HM1a1* and *HM1a2* could be replaced by the expression on the right-hand sides of constraints (39) and (40), thus making the variables *HM1a1* and *HM1a2* as well as the constraints (39) and (40) obsolete. In the above formulation a good documentation was favored, which makes the problem size bigger than it has to be.

3 THE SIMPLEX METHOD

Being not only the traditional and best-known method but also the standard solution technique for the solution of linear programming problems, we devote this chapter entirely to the simplex method.

The chapter is subdivided into two sections. The first section introduces the graphical solution method and discusses some resulting properties and special cases. The second section derives the simplex method, the standard technique for the solution of linear programming problems. It then returns to the special cases and examines them algebraically.

3.1 Graphical Concepts

This section first describes how to plot the feasible set as well as the objective function of a linear programming problem with two variables, before putting things together in the graphical solution method. The section ends with a discussion of some special cases that may occur during the execution of the solution technique. While any graphical approach is limited to no more than three variables, its discussion is nonetheless valuable, as it provides insight into the problem and allow us to estimate the effects of changes in the individual parameters in the context of postoptimality analyses of the type to be discussed in detail in Chapter 6.

3.1.1 The Graphical Solution Technique

The graphing of the feasible set on the basis of any number of given $\leq$, $=$, and $\geq$ constraints has already been discussed in Section A.2. Consider now the objective function and, without loss of generality, assume that the given objective function is Max $z = \sum_{j=1}^{n} c_j x_j$. For any given value of z, e.g., $z = z_0$, this function can be

written as $\sum_{j=1}^{n} c_j x_j = z_0$, which is a constraint with a right-hand side value of z_0, which can be represented by a hyperplane in $\mathbb{R}^n$. For any value of the objective function $z_1 \neq z_0$, we obtain a different hyperplane, which is parallel to the one obtained for z_0. Each of these hyperplanes consists of points that are all equally good in the sense that they have the same value of the objective function. For that reason, they are frequently referred to as *isoprofit lines* or *isocost lines,* depending on the objective function, or sometimes as *contour lines*. (Since the term "lines" is only valid in two-dimensional problems, it might be replaced by "hyperplanes" in a more general treatment.) Given a maximization problem, the objective function will then attempt to find points on the contour line that belongs to the highest possible objective value. (For minimization problems, "highest possible" is replaced by "lowest possible.") The only feature that restricts choosing points with arbitrary high objective values are the constraints. The objective is then to find a point in the feasible set that is located on the contour line with the highest (lowest for minimization) possible value of the objective function.

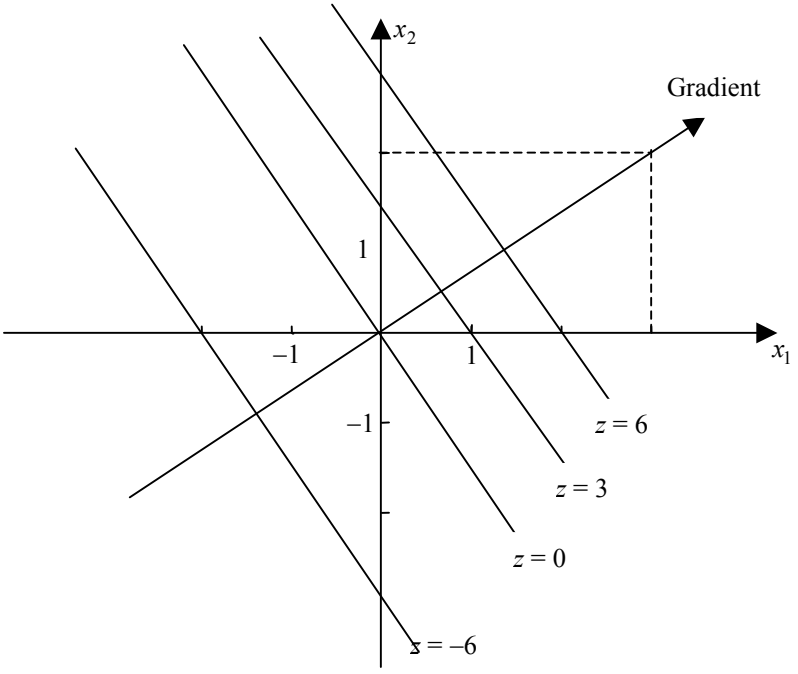


Figure 3.1

This search can be accommodated by plotting the unique unidimensional ray $\mathbf{x} = \lambda\mathbf{c}$, $\lambda \in \mathbb{R}$, which is orthogonal to the isoprofit hyperplanes $\mathbf{cx} = z$, $z \in \mathbb{R}$. This ray

is called the *direction* or the *gradient of the objective function* and it indicates in which direction the hyperplanes **cx** = z have to be shifted, in order to achieve a better (higher for maximization and lower for minimization problems) value of the objective function. Notice that the gradients of Max z_1 = **cx** and Min z_2 = **cx** are diametrically opposed. As an illustration, consider the following

Example: Consider the objective function Max $z = 3x_1 + 2x_2$. The graph in Figure 3.1 displays the gradient of this objective function as well as isoprofit lines for z = –6, 0, 3 and 6, respectively. The arrow points in the direction in which the objective function values improve.

Example 2: Consider the task of plotting the gradients of the following objective functions:

(1) Max $z = x_1 + 2x_2$ or, equivalently, Min $-z = -x_1 - 2x_2$
(2) Max $z = -3x_1 + x_2$ or, equivalently, Min $-z = 3x_1 - x_2$
(3) Max $z = -3x_1 - 2x_2$ or, equivalently, Min $-z = 3x_1 + x_2$, and
(4) Max $z = x_1 - x_2$ or, equivalently, Min $-z = -x_1 + x_2$.

Figure 3.2 shows the gradients of all four objective functions.

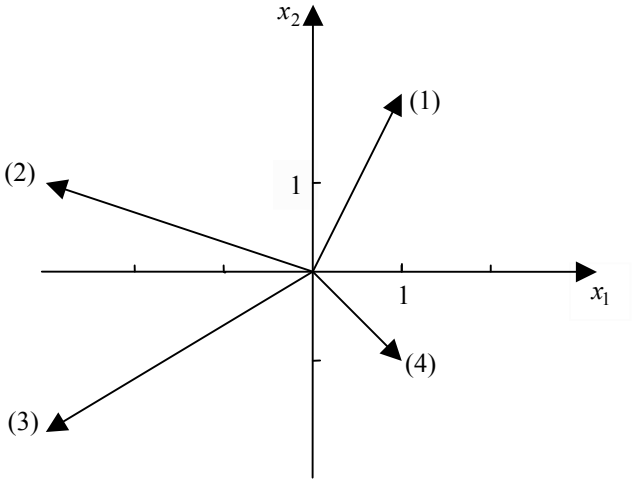


Figure 3.2

At this point, we will introduce the constraints and describe the graphical solution technique. Consider the graph in Figure 3.3 where the shaded area denotes the polytope given by the constraints *I*, *II* and *III* and in which the gradient of the objective function as well as the contour lines for some objective function values are also displayed.

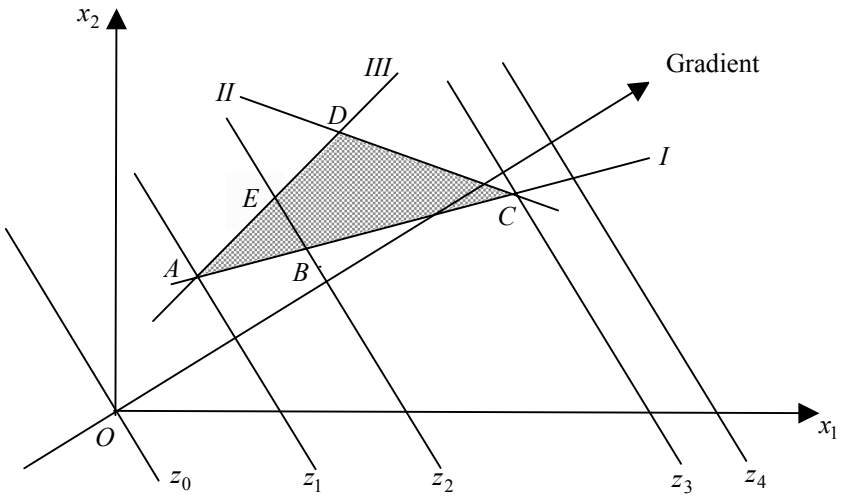


Figure 3.3

According to the gradient of the objective function, $z_4 > z_3 > z_2 > z_1 > z_0$. Suppose the objective is to maximize profit and start with a profit level of z_0. All points on the z_0 contour line represent an equal profit but none of these points is feasible. Increasing the profit level to z_1 we find that A is the only feasible point on this level. A further increase to z_2 indicates that points B and E as well as all points on the line segment between B and E are not only feasible but are also all on the same profit level, i.e., all of the points are considered equally good. Increasing the profit level even further to z_3, we see that only one point, namely C, is feasible on this level. Any further increase in the profit level results in a set of points on a hyperplane with a profit level of, say, z_4, all of whose points have a higher profit level than point C, but none of those points is feasible. We can conclude that point C represents the unique optimal solution. In simple words, the contour hyperplanes are shifted in the direction of the objective function, until one of them touches the last point of the given polytope; this point is optimal. In the above example we have seen that point C is optimal, and it is now our task to determine the exact coordinates of that optimal point.

Since C is uniquely determined by the intersection of the hyperplanes bordering the halfspaces of constraints *I* and *II*, we have a system of two equations (*I* and *II*) in the two variables x_1 and x_2 which has a unique solution, consisting of the exact $(\bar{x}_1, \bar{x}_2)$ coordinates of the optimal point C. Replacing x_1 and x_2 by the values $\bar{x}_1$ and $\bar{x}_2$ in the objective function results in the optimal z-value $\bar{z}$ and the problem is completely solved. This procedure can be summarized as follows:

The Graphical Solution Technique

Step 1: Graph all constraints and determine the feasible set.

Step 2: Graph the gradient of the objective function.

Step 3: Shift the contour lines into the direction of the gradient of the objective function until the last feasible point is reached. This point $\bar{\mathbf{x}}$ is optimal.

Step 4: The constraints of the set of hyperplanes that intersect at $\bar{\mathbf{x}}$ are satisfied as equations at optimum. Solve that system of simultaneous linear equations, e.g., with the Gauss-Jordan pivoting method (see Procedure A.19 in Section A.2). The coordinates of $\bar{\mathbf{x}}$ constitute an optimal solution.

Step 5: Determine the value of the objective function at optimum as $\bar{z} = \mathbf{c}\bar{\mathbf{x}}$.

The graphical solution method described above clearly indicates that no interior point can be optimal, so that any optimal solution must be located on the boundary of the feasible set. We can then prove

Lemma 3.1: (Dantzig, 1951): Assume that the feasible set is nonempty and bounded. Then at least one optimal solution is located at an extreme point.

Proof: Assume for now that $\bar{\mathbf{x}}$ is a unique optimal solution and suppose that $\bar{\mathbf{x}}$ is not at an extreme point. Due to the compactness and convexity of the feasible set, there will then exist feasible extreme points $\mathbf{x}^1$, $\mathbf{x}^2$, so that $\bar{\mathbf{x}}$ can be expressed as a linear convex combination of $\mathbf{x}^1$ and $\mathbf{x}^2$ neither of which is optimal, i.e., $\bar{\mathbf{x}} = \lambda\mathbf{x}^1 + (1-\lambda)\mathbf{x}^2, \lambda \in]0,1[$. Since $\bar{\mathbf{x}}$ is by assumption optimal while $\mathbf{x}^1$ and $\mathbf{x}^2$ are not, $\mathbf{c}\bar{\mathbf{x}} > \mathbf{c}\mathbf{x}^1$ and $\mathbf{c}\bar{\mathbf{x}} > \mathbf{c}\mathbf{x}^2$. Now $\mathbf{c}\bar{\mathbf{x}} > \mathbf{c}\mathbf{x}^1$ can be written as $\mathbf{c}[\lambda\mathbf{x}^1 + \mathbf{x}^2 - \lambda\mathbf{x}^2] - \mathbf{c}\mathbf{x}^1 > 0$; which, in conjunction with the assumption that $\lambda < 1$ (if $\lambda = 1$, then $\bar{\mathbf{x}}$ coincides with the extreme point $\mathbf{x}^1$, thus violating the assumption that it is not an extreme point), result in $\mathbf{c}\mathbf{x}^1 < \mathbf{c}\mathbf{x}^2$. An equivalent argument can be applied to $\mathbf{c}\bar{\mathbf{x}} > \mathbf{c}\mathbf{x}^2$ which results in $\mathbf{c}\mathbf{x}^1 > \mathbf{c}\mathbf{x}^2$, an obvious contradiction. This process can now be repeated for all optimal points. It is impossible that all of these points are non-extreme points, as that would lead to a contradiction in all cases. This proves the lemma. □

Example: Consider the following linear programming problem.

P: Max $z = 2x_1 + x_2$

s.t. $x_1 + x_2 \geq 1$ (*I*)

$3x_1 + 4x_2 \geq 12$ (*II*)

$$
\begin{aligned}
x_1 - x_2 &\le 2 && (III)\\
-2x_1 + x_2 &\le 2 && (IV)\\
x_1 &\ge 0 && (V)\\
x_2 &\ge 0 && (VI)
\end{aligned}
$$

The graph of this problem is shown in Figure 3.4, where, as usual, the shaded area represents the polytope and the dashed lines indicate contour hyperplanes. Using the graphical solution technique we find that point C is optimal. Since C is determined by the intersection of the bordering hyperplanes of halfspaces *II* and *III*, the system of simultaneous linear equations to be solved is $3x_1 + 4x_2 = 12$ and $x_1 - x_2 = 2$. The result is the optimal solution $\bar{\mathbf{x}} = \left(2\frac{6}{7}, \frac{6}{7}\right)$ with the objective value $\bar{z} = 6\frac{4}{7}$.

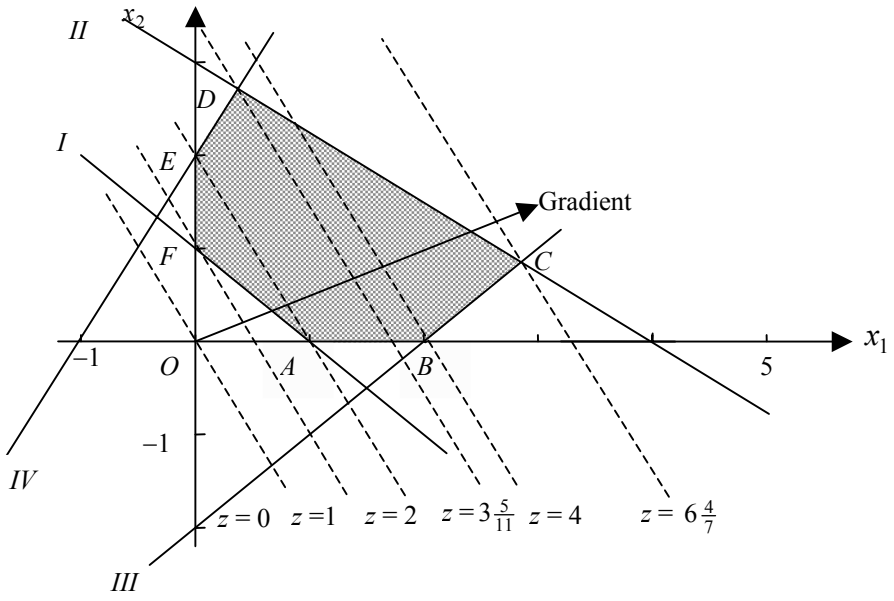


Figure 3.4

At this point it is beneficial to look at techniques that allow us to determine an optimal solution, even if a graphical representation of the given problem is not available. At any given feasible point, we define an *improving feasible direction* as a direction, in which

(1) a move with a short step length will preserve feasibility, and
(2) the value of the objective function improves (i.e., increases for maximization and decreases for minimization problems).

An improving feasible direction method that starts at an extreme point of a polytope and moves along an improving feasible direction along the boundary of the polytope to an adjacent extreme point, is called a (*phase 2*) *simplex path*.

We will demonstrate a simplex path by virtue of the example shown in Figure 3.4. Arbitrarily start with F as the first feasible point.

From F, we can move to either A or E, since both are adjacent to the present solution F, both are feasible, and both are located on higher contour lines than F. Here, we arbitrarily choose point E, which, again, has two neighbors, *viz*., F and D. We cannot move back to F since it is located on a lower profit level; hence we move to D which, in turn, has the two neighbors E and C. From point D we must move to C, as the only other alternative, E, is located at a lower contour line. Consider now the situation at point C. Both of point C's neighboring extreme points B and D are located at lower profit levels. Even if we were not restricted to extreme points, it is apparent that there exists no feasible point anywhere in the neighborhood of point C that has a higher objective function value than C. We can therefore conclude that point C is optimal. This is the way the simplex method proceeds.

In the above example, there are two possible simplex paths from point F, *viz*., $F - E - D - C$ and $F - A - B - C$. Which of the two paths the simplex method chooses will depend on the finetuning of its parameters. Many attempts were made in the 1960s and 1970s to find one of the shortest paths, as this will certainly minimize the computational effort. However, none of these attempts proved successful. One of the reasons is that none of the methods has foresight much beyond what happens when moving from one extreme point to an adjacent extreme point (other than actually making the moves), a feature required to find short paths between the present solution and the optimal point.

Another set of simplex paths will be described in the following three-dimensional

Example: Consider the following linear programming problem:

$$
\begin{aligned}
\text{P: Max } z = {} & 4x_1 + x_2 + 3x_3 \\
\text{s.t.} \quad & x_1 \leq 3 \\
& x_2 \leq 5 \\
& x_3 \leq 2 \\
& x_1 + x_2 + 2x_3 \leq 10 \\
& x_1, \; x_2, \; x_3 \geq 0.
\end{aligned}
$$

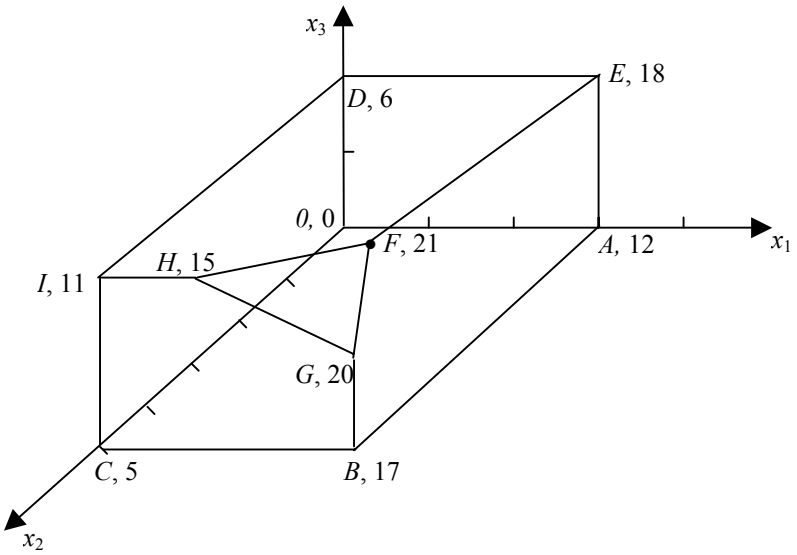


Figure 3.5

The problem describes a polyhedron with the ten extreme points $O = (0, 0, 0)$, $A = (3, 0, 0)$, $B = (3, 5, 0)$, $C = (0, 5, 0)$ $D = (0, 0, 2)$, $E = (3, 0, 2)$, $F = (3, 3, 2)$, $G = (3, 5, 1)$, $H = (1, 5, 2)$ and $I = (0, 5, 2)$. The feasible set of the problem is shown in Figure 3.5 in which the numbers next to the extreme points are their values of the objective function. Starting with point O, we can move to one of its three neighbors A, C or D. Suppose we move to D whose neighboring extreme points are O, E and I; however, since the objective value must increase, we can only move to E or I. Arbitrarily select I as the next point. Among point I's three neighbors C, D and H, only a move to H is possible. From H, a move to F or G is possible, moving back to I is "illegal" as it decreases the value of the objective function. Suppose that a move to point F is selected and, since none of its neighbors E, H and G has a better z-value, F is optimal. Even in this small model, there are already eight "legal" simplex paths form O to F, *viz*., (O, A, B, G, F), (O, A, E, F), (O, C, B, G, F), (O, C, I, H, G, F), (O, C, I, H, F), (O, D, E, F), (O, D, I, H, G, F), and (O, D, I, H, F). These paths include between four and six extreme points. In practice, it would be very beneficial to know beforehand which path from the starting point to the optimal solution includes the smallest number of intermediate extreme points, as this would minimize the computational effort required to solve the problem. Since, however, only local information is available at the individual extreme points, such information is not available.

Consider now the example in Figure 3.6. Assume that the initial point is B at a level of the objective function of z_0. Point B has the two neighbors A and D. Both are feasible and both have higher values of the objective function, making either move feasible. Suppose now that we choose to move to point A. Here, both

adjacent points *B* and *C* have lower values of the objective function than *A*, leading us to the conclusion that *A* must indeed be an optimal solution. However, simple inspection reveals that point *D* represents a unique optimal solution.

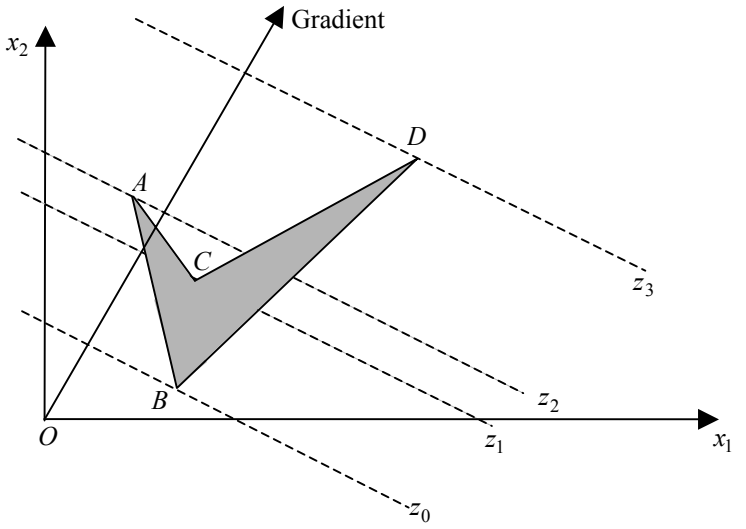


Figure 3.6

This difficulty requires us to define the type of problems for which the feasible direction approach works properly.

Definition 3.2: A *local optimum* for a maximization problem on a set *S* is defined as a point $\hat{\mathbf{x}} \in S$, such that for some arbitrarily small $\varepsilon > 0$ and all $\tilde{\mathbf{x}} \in S$, $\|\hat{\mathbf{x}} - \tilde{\mathbf{x}}\| \leq \varepsilon$ implies that $\mathbf{c}\tilde{\mathbf{x}} \leq \mathbf{c}\hat{\mathbf{x}}$. A *global optimum* $\bar{\mathbf{x}} \in S$ is a point, such that $\mathbf{c}\bar{\mathbf{x}} \geq \mathbf{c}\tilde{\mathbf{x}} \quad \forall\, \tilde{\mathbf{x}} \in S$.

Using this definition, it is possible to prove

Lemma 3.3: In a problem with a convex feasible set and a linear objective, each local optimum is a global optimum.

We can also state (here without proof)

Lemma 3.4: An improving feasible direction search that moves from one extreme point to an adjacent extreme point of a closed polyhedron finds a local optimum.

Lemmas 3.3 and 3.4 imply that the neighborhood search technique that results in

what we call a simplex path will indeed find a global optimum of a linear programming problem. This is not a contradiction to the example in Figure 3.6, as the feasible set shown therein is not convex; that set could not be generated as the intersection of linear halfspaces and hyperplanes as is the case in linear programming.

In this context we would like to mention the Hirsch conjecture of 1957, which states that it is generally possible to move from any extreme point to any other in at most m steps, where m is the number of constraints. Klee und Walkup (1967) proved the Hirsch conjecture (Hirsch, 1957) for problems with $n - m \leq 5$, where n is the number of variables, and disproved it for problems with unbounded polytopes.

3.1.2 Four Special Cases

Below, we will discuss four special cases that may occur in the optimization process. The first two cases are related to issues related to the formulation of the model, meaning that the model has to go back to the drawing board. We will offer further comments when we discuss these cases in some detail. The last two cases are of a technical nature. They may be of only passing interest, or they may cause problems in the workings of the algorithm. Again, more details are offered below.

Case 1: No feasible solutions exist, if the intersection of all given halfspaces and/or hyperplanes is empty. Clearly, this depends exclusively on the constraints and indicates that some *constraints are too tight*, and the model has to be reformulated before any optimization can resume.

Example: Consider the following set of constraints:

$$\begin{aligned} x_1 &\leq 2 && (I)\\ x_2 &\leq 1 && (II)\\ 2x_1 + 5x_2 &\geq 10 && (III)\\ x_1, \quad x_2 &\geq 0. \end{aligned}$$

As Figure 3.7 clearly indicates, the feasible set is empty. In this small example, this could also be seen in the set of constraints where constraint (*I*) implies that $2x_1 \leq 4$, constraint (*II*) implies that $5x_2 \leq 5$, so that constraints (*I*) and (*II*) together have to satisfy $2x_1 + 5x_2 \leq 9$, which is a direct contradiction to constraint (*III*). One question is then which of the constraints causes the infeasibility. However, this is the wrong question. In the above example, it is apparent that any two of the three constraints allow an infinite number of feasible solutions, while adding the remaining constraint causes the infeasibility. In other words, the nonexistence of feasible solutions is caused by the incompatibility of a group of constraints. Most software codes will display the group of constraints that cause the infeasibility, which is a great help to modelers when redesigning the model.

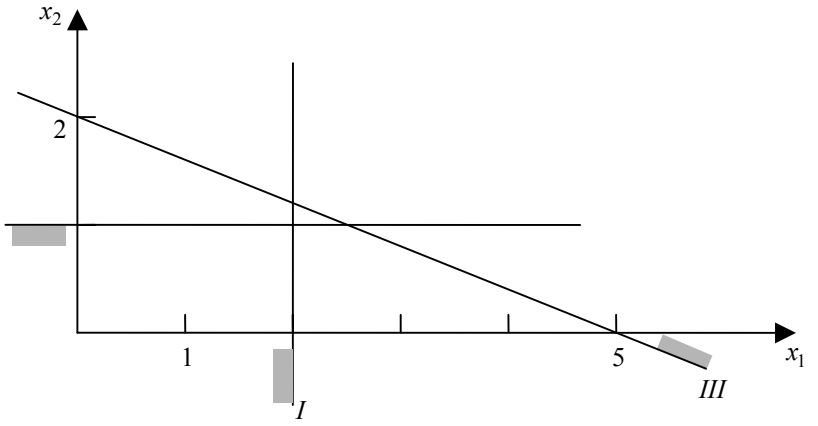


Figure 3.7

In order to restore feasibility, it is necessary for the modeler to *loosen up* some of the constraints that are involved in causing the infeasibility (such a set is often identified by the optimizer). Loosening up constraints means allowing a greater degree of leeway. For instance, a resource constraint may be loosened up by trying to find ways of acquiring more resources, while a demand constraint that requires a certain minimum amount to be shipped to a customer may be loosened up by requesting from the customer a smaller required quantity. Simply speaking, $\leq$ constraints are loosened by increasing the right-hand side value, while $\geq$ constraints are loosened by decreasing the right-hand side value. Equations—this holds for formulations in general and not just cases in which feasible solutions do not exist—should be avoided whenever possible, as they dramatically restrict the set of feasible solutions. When loosening up constraints, it is usually advantageous to modify some, rather than a single, right-hand side values of the constraints that cause the infeasibility. The reason is that restricting ourselves to changing only a single right-hand side value may require a massive change, which could be difficult to implement, while working on multiple constraints it may be possible to restore feasibility by a number of smaller changes to the existing parameters.

Case 2: Unbounded "optimal" solutions exist, if the polytope is unbounded and the gradient of the objective function is directed into the cone given by rays that are orthogonal to the diverging hyperplanes.

Example: Consider the polytope given by the constraints

$$\begin{aligned} -2x_1 + x_2 &\leq 2 \qquad (I)\\ x_1 - 3x_2 &\leq 3 \qquad (II)\\ x_1, \quad x_2 &\geq 0 \end{aligned}$$

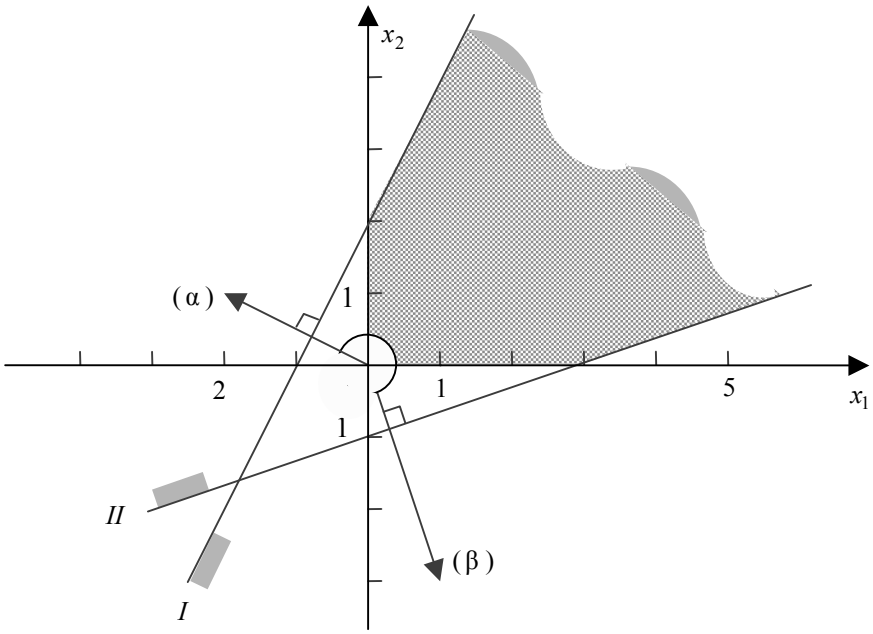


Figure 3.8

Every objective function between (α): Max $z = -2x_1 + x_2$ and (β): Max $z = x_1 - 3x_2$ pointing in the northeast direction results in unbounded optimal solutions, whereas every objective function between (α) and (β) pointing in the southwest direction results in a finite optimal solution (either (0, 2), (0, 0) or (3, 0)). If the objective function is (α), then there are alternative optimal solutions $\bar{\mathbf{x}} = (M, 2M + 2)$ for any $M \geq 0$, all have a value of the objective function $\bar{z} = -2M + 2M + 2 = 2$. Although the values of both variables may be unrestricted, the z-values is bounded and we do not speak of unbounded optimal solutions. The same is true for objective (β) where $\bar{\mathbf{x}} = (3M + 3, M)$ produces $\bar{z} = 3$ for all $M \geq 0$. The above discussion reveals that the existence of unbounded optimal solutions depends on the constraints as well as on the objective function. Whenever unbounded "optimal" solutions are encountered, an error has been made in the formulation. In particular, some of the relevant constraints of the model may have been forgotten, so that the existing formulation is "too loose."

Case 3: Dual degeneracy occurs at an extreme point, if at least one of its adjacent extreme points has the same value of the objective function. If dual degeneracy occurs at optimal extreme point, then at least one *alternative optimal solution* exists. For the occurrence of dual degeneracy it is necessary and sufficient that the

gradient of the objective function is orthogonal to the (boundary) hyperplane of a nonredundant constraint. The occurrence of dual degeneracy depends on the constraints as well as on the gradient (or direction) of the objective function.

Example: Consider the following linear programming problem

$$\begin{aligned} \text{P: Max } z = \; & x_1 + x_2 \\ \text{s.t.} \quad & x_1 + x_2 \geq 1 \qquad (I) \\ & x_1 + x_2 \leq 2 \qquad (II) \\ & x_1 \,, \;\; x_2 \geq 0. \end{aligned}$$

Figure 3.9 indicates that all four extreme points are dual degenerate: Points A and D are neighbors and have the same value of the objective function, and points B and C are also neighbors with the same z-value. Furthermore, since the extreme points B and C are both optimal, both are alternative optimal solutions. In addition, all points on the line segment between B and C are also optimal.

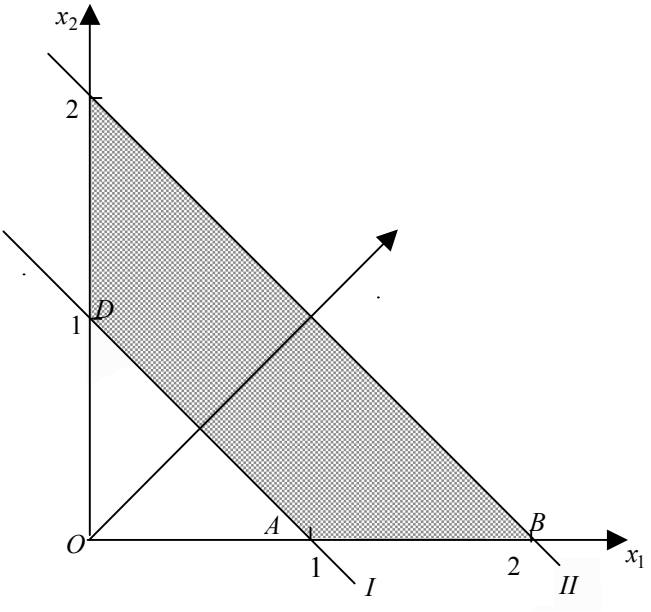


Figure 3.9

Case 4: Primal degeneracy occurs if more than n hyperplanes intersect at one point of $\mathbb{R}^n$ (which makes the point "overdetermined"). Primal degeneracy depends exclusively on the constraints and not on the objective function.
Example: Consider the polytope defined by the following constraints:

$$\begin{aligned} 4x_1 + 2x_2 &\leq 9 && (I)\\ 2x_1 + 3x_2 &\leq 6 && (II)\\ 6x_1 + 5x_2 &\leq 15 && (III)\\ x_1, \quad x_2 &\geq 0 \end{aligned}$$

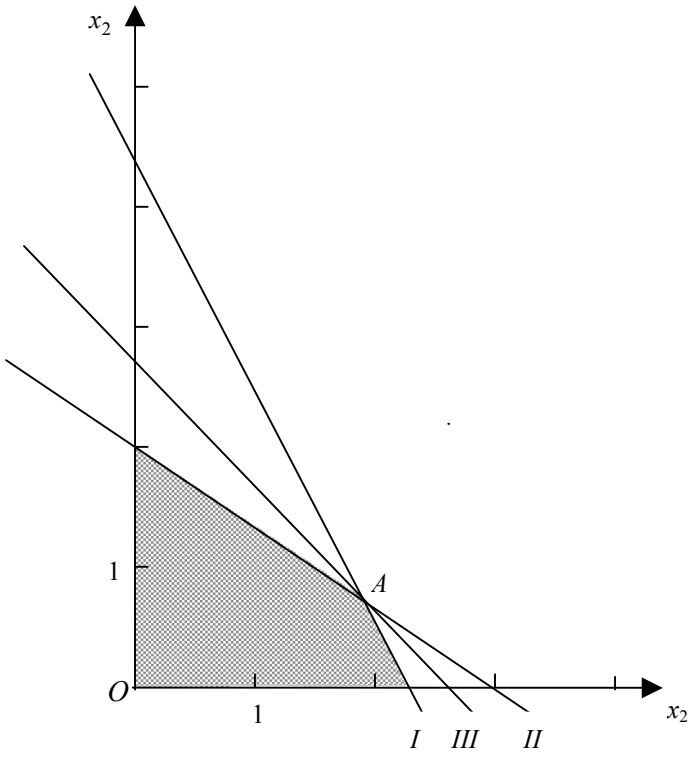


Figure 3.10

The graph in Figure 3.10 reveals that primal degeneracy occurs at the point $A = (1\frac{7}{8}, \frac{3}{4})$ at which all three constraints are satisfied as equalities. (Note that constraint *III* is redundant. In general, however, primal degeneracy can exist in the absence of redundancy as well, as exemplified by a pyramid with four sloping sides. Its apex is primal degenerate, but there is no redundancy).

3.2 Algebraic Concepts

In this section we will describe an algebraic technique which can be used for the solution of any mathematical programming problem that has been reduced to a linear programming problem, e.g., by virtue of one of the transformations to be described in Chapter 8. Once the general technique is described an illustrated by an example, we will revisit the four special cases that have been discussed above from a graphical point of view.

3.2.1 The Algebraic Solution Technique

For the time being, we will restrict ourselves to problems in *canonical form* with nonnegative right-hand sides, i.e., problems of the type

$$\text{P: Max } z = \sum_{j=1}^{n} c_j x_j$$

$$\text{s.t. } \sum_{j=1}^{n} a_{ij} x_j \le b_i \ , \ \ i = 1,...,m \ ,$$

$$x_j \ge 0, j = 1,\ldots,n$$

or, in matrix notation

$$\text{P: Max } z = \mathbf{cx}$$

$$\text{s.t. } \mathbf{Ax} \le \mathbf{b}$$
$$\mathbf{x} \ge \mathbf{0},$$

where $b_i \ge 0 \ \forall \ i = 1, \ldots, m$, or $\mathbf{b} \ge \mathbf{0}$ in matrix notation. This formulation does not include constraints such as $\mathbf{a}_{i\bullet}\mathbf{x} \ge b_i$ and/or $\mathbf{a}_{i\bullet}\mathbf{x} = b_i$. For any of these problems, the origin $\mathbf{x} = (0, 0, \ldots, 0)^T$ is feasible, so that the existence of at least one optimal solution is assured. In other words, for now, we are dealing only with the second phase of a general optimization problem.

Adding slack variables $S_1, S_2, \ldots, S_m$ as prescribed in Chapter 1 leads to the formulation in *standard form*

$$\text{P: Max } z = \sum_{j=1}^{n} c_j x_j$$

$$\text{s.t.} \qquad \sum_{j=1}^{n} a_{ij} x_j + S_i = b_i, \ i = 1,...,m$$

$$x_j \ge 0, j = 1, \ldots,n$$
$$S_i \ge 0, i = 1, \ldots,n,$$

or in matrix form as

$$\begin{aligned} &\text{P: Max } z = \mathbf{cx} \\ &\text{s.t.} \quad \mathbf{Ax} + \mathbf{IS} = \mathbf{b} \\ &\qquad\quad \mathbf{x} \quad\; \geq \mathbf{0} \\ &\qquad\quad\;\; \mathbf{S} \geq \mathbf{0}. \end{aligned}$$

Since exactly one slack variable has been added to each of the m constraints, we now have a total of $(n + m)$ variables, m constraints and $(n + m)$ nonnegativity constraints for a total of $2m + n$ constraints. A simplex tableau that includes all constraints (except for the nonnegativity constraints that are handled implicitly) is shown below. Note that the scalar z_0 on the right-hand side of the objective function denotes the constant associated with the solution, in which all decision variables assume a value of zero. For example, z_0 would be a fixed cost or a basic revenue that is obtained regardless of the actual solution. The general appearance of a simplex tableau is shown in Table 3.1.

Table 3.1

T:	$\mathbf{x}$	$\mathbf{S}$	1	
m rows	$\mathbf{A}$	$\mathbf{I}$	$\mathbf{b}$	constraints
1 row	$-\mathbf{c}$ (n columns)	$\mathbf{0}$ (n columns)	z_0 (1 column)	objective

Definition 3.5: A tableau is said to include a *basic solution* if it contains the m unit column vectors $\mathbf{e}_{\bullet i}$, $i = 1,\ldots, m$. The variables heading these m columns are called the *basic variables*. The other variables are called *nonbasic variables*.

Definition 3.6: A basic solution is called a *basic feasible solution* (*BFS*) if the current values of all variables are nonnegative. A basic variable x_j under which appears a unit vector $\mathbf{e}_{\bullet i}$ is said to be *in the basis* in the i-th row, and its current value is $x_j = b_i$. All nonbasic variables have a current value of zero.

The initial basis shown in the tableau above consists of all m slack variables, while the remaining n decision variables are nonbasic. The present values of the variables are $x_j = 0$, j=1, …, n and $S_i = b_i$, $i = 1, \ldots, m$, and the value of the objective function is presently z_0. This value may include any fixed costs or profits as such a constant will not influence the optimization and the optimal solution.

The simplex method by Dantzig has been the method of choice for the last sixty years and is still (by far) the method used for most optimization purposes, and it proceeds as follows. The method first determines whether or not the current solution is optimal. If it is, the procedure terminates; if not, it selects a nonbasic

variable x_j, which, by definition, is currently at the zero level, and which increases the value of the objective function if its own value increases. Every nonbasic variable with a negative entry in the objective function row has this property.

If not, we will change the present basis by designating the column of a present nonbasic variable as the *entering variable*, which will enter the basis in this step, and the column of a variable that is presently in the basis as the *leaving variable*, i.e., the variable that will leave the basis in this step. Given that basis change, the next basis will be a neighbor of the present basis in the sense that the two bases are identical except for a single variable.

For simplicity of the exposition, assume now that the variables are $x_1, x_2, \ldots, x_n, x_{n+1}, x_{n+2}, \ldots, x_{n+m}$, i.e., they include the problem variables as well as the slack variables. The negative value of the *indicator* or *reduced cost* $-c_j$ of a variable x_j will then indicate by how many units the value of the objective function will increase if the value of the variable x_j were to increase by one. It is apparent that if $-c_j \geq 0 \ \forall \ j$, then the present solution is optimal as no further increase in the objective function is possible.

Assume now that the present solution is not yet optimal. Then there exists at least one $-c_s < 0$ and we will attempt to increase the value of the variable x_s which is only possible if we introduce the variable into the basis. Clearly, the increase of any variable cannot be done in an isolated way, but will require changes to a number of other variables. To simplify matters, assume that $x_i = b_i$, $i = 1,\ldots, m$, i.e., x_i is a basic variable in the i-th row. The column vector under the variable x_s is $[a_{1s}, a_{2s},\ldots, a_{is},\ldots, a_{ms}]^T$, which indicates that if x_s were to be increased by some $\varepsilon > 0$, the right-hand side values will change to $[b_1-\varepsilon\ a_{1s}, b_2-\varepsilon\ a_{2s},\ldots, b_i-\varepsilon\ a_{is},\ldots, b_m-\varepsilon\ a_{ms}]^T$, i.e., the values of the basic variables change to $x_i = b_i - \varepsilon\ a_{is}$, $i = 1,\ldots, m$. Since the values of all variables have to remain nonnegative, we require that $b_i - \varepsilon a_{is} \geq 0 \ \forall \ i = 1, \ldots, m$ with $a_{is} > 0$ (as for $a_{is} \leq 0$ the new values can never become negative). Simple algebra leads to the condition $\varepsilon \leq \dfrac{b_i}{a_{is}}, i = 1,..,m$ with $a_{is} > 0$ or simply $\varepsilon \leq \min\limits_{i=1,\ldots,m} \left\{ \dfrac{b_i}{a_{is}} : a_{is} > 0 \right\}$. Suppose now that $b_i > 0$, $i = 1,\ldots, m$ and $\min\limits_{i=1,\ldots,m} \left\{ \dfrac{b_i}{a_{is}} : a_{is} > 0 \right\}$ is uniquely determined by, say, $\dfrac{b_r}{a_{rs}}$. If we were to choose some $\varepsilon \in \left] 0; \dfrac{b_r}{a_{rs}} \right[$, then $x_s > 0$ and $x_i > 0$, $i = 1,\ldots, m$; hence a total of $(m + 1)$ variables out of the total $m + n$ variables would have a strictly positive value. Following Definition 3.5, this is not a basic solution, since in order for the $(m+1)$ variables to be positive, they have to be basic variables, and a basic solution cannot include more than m basic variables. However, if we were to

choose $\varepsilon = \dfrac{b_r}{a_{rs}}$, then again $x_s > 0$ but $x_i > 0$, $i = 1,\ldots, m \;\forall\; i \neq r$ since $x_r = b_r - \varepsilon a_{rs}$ $= b_r - \dfrac{b_r}{a_{rs}} a_{rs} = 0$, so that now no more than m variables have a positive value. Now all n nonnegativity constraints as well as the given m constraints are satisfied, making the new solution a basic feasible solution.

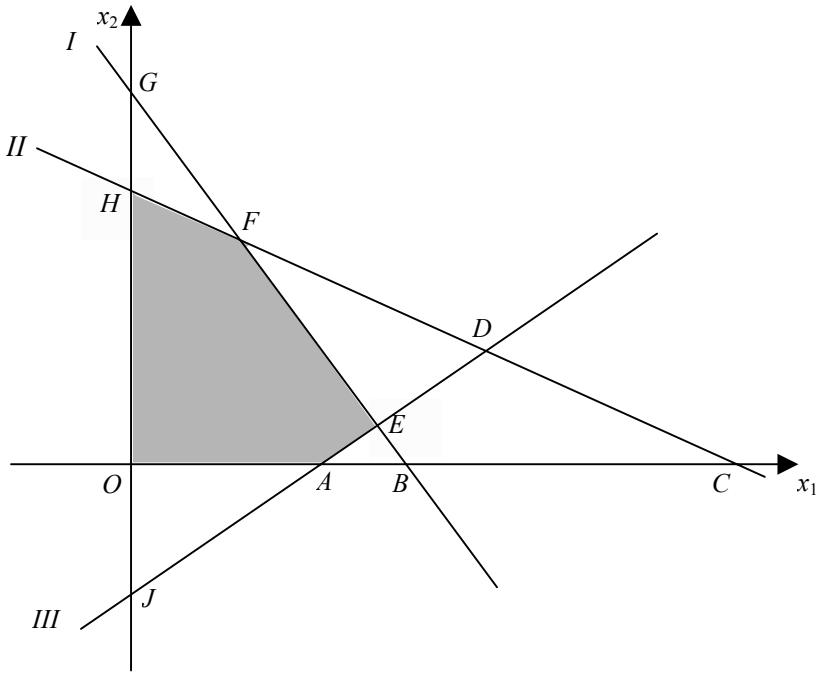


Figure 3.11a

Let us now briefly clarify the relationships between the algebraic concepts of basic solutions and basic feasible solutions, see Definitions 3.5 and 3.6, and their respective geometric counterparts. Every basic solution is represented by one and only one basic point; however, the same basic point may be the representation of more than one basic solution. Similarly, every basic feasible solution is represented by one and only one extreme point; however, the same extreme point may be the representation of more than one basic feasible solution. This implies that the number of basic solutions is never less than the number of basic points; similarly, the number of basic feasible solutions is never less than the number of extreme points. The number of basic points is however unrelated to the number of basic feasible solutions. This is illustrated in Figures 3.11a and b.

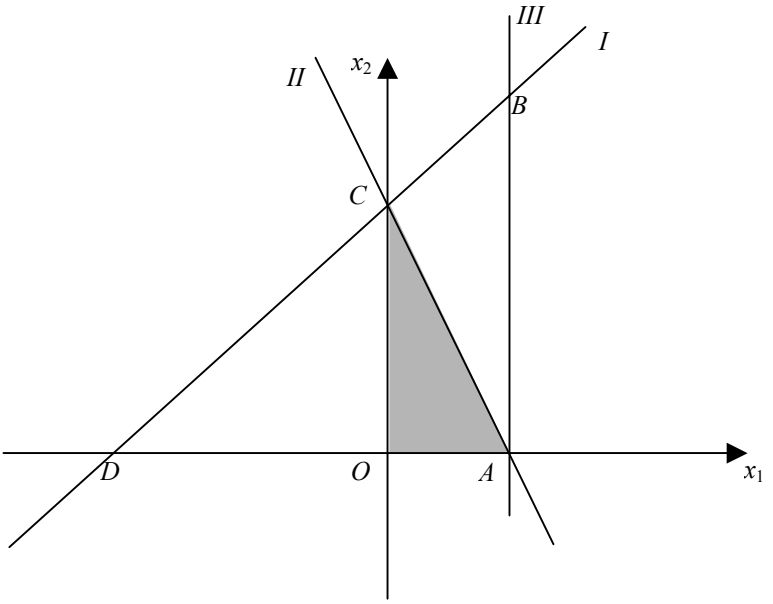


Figure 3.11b

The problems graphed in Figures 3.11a and 3.11b each consist of $n = 2$ nonnegative variables and $m = 3$ structural constraints *I*, *II*, *III*. In Figure 3.11a there are ten basic points corresponding to ten basic solutions and five basic feasible points corresponding to five basic feasible solutions. In contrast, the problem in Figure 3.11b includes five basic points corresponding to as many as nine basic solutions (this can be seen by shifting the hyperplane *II* slightly to the northeast, resulting in the decomposition of points *A* and *C* into three distinct points each). Figure 3.11b also contains three basic feasible points. By the same argument, there are five basic feasible solutions.

Consider now the following: a basis in the $[m \times (n + m)]$-dimensional tableau is an $[m \times m]$-dimensional submatrix, i.e., a collection of m out of $(n + m)$ columns. Clearly, there exist $\binom{n+m}{m}$ of these $[m \times m]$-dimensional submatrices, while the number of bases may be less since not every $[m \times m]$-dimensional submatrix has full rank as required by a basis. Knowing that at least one of the optimal solutions must be located at an extreme point, we could enumerate all $[m \times m]$-dimensional submatrices $\mathbf{B}$ and solve the resulting $\binom{n+m}{m}$ systems of simultaneous linear equations $\mathbf{x_B} = \mathbf{B}^{-1}\mathbf{b}$. All submatrices for which either $\mathbf{B}^{-1}$ does not exist or for which $\mathbf{x_B}$ is not nonnegative are discarded; for all remaining bases $z = \mathbf{cx_B}$ is

calculated and the basis with the highest $\mathbf{c}\begin{bmatrix}\mathbf{x_B}\\ \mathbf{x_N}\end{bmatrix}$ value determines the optimal solution. While valid, this procedure is not practical, because even for small problems, say $n = 10$ problem variables and 10 constraints, as many as $\binom{10+10}{10} =$ 184,756 systems of simultaneous linear equations may have to be set up and solved. With this in mind, we can now fully appreciate the much more efficient simplex algorithm which, following simplex folklore, requires an average of $\frac{3}{2}m$ iterations, where m is the number of constraints. A formal description of Phase 2 is provided below. If not otherwise stated, c_j refers to the objective function coefficient of the j-th variable *as it appears in the tableau*, the vector $\mathbf{x}$ includes *all* variables, and a_{ij} is the coefficient of x_j in the i-th row, no matter if it belongs to a decision, slack or any other variable. Note that all coefficients c_j, a_{ij} and b_i are redefined in every iteration; if we refer to one of them we always refer to its current value.

The Primal Simplex Algorithm: Phase 2

Step 1: Is $c_j \geq 0, j = 1,\ldots, n$?
If yes: Stop, the current solution is optimal.
If no: Go to Step 2.

Step 2: Select any nonbasic variable x_s with $c_s < 0$ as entering variable. The s-th column is called the *pivot column*.

Step 3: Is there any positive element $a_{is} > 0$, $i = 1,\ldots,m$ in the pivot column?
If yes: Go to Step 4.
If no: Stop, there are unbounded "optimal" solutions.

Step 4: Select the r-th row as *pivot row*, such that

$$\frac{b_r}{a_{rs}} = \min_{i=1,\ldots,m}\left\{\frac{b_i}{a_{is}} : a_{is} > 0\right\}$$

The variable which is in the basis in the r-th row leaves the basis. The element $a_{rs} > 0$ is the pivot.

Step 5: Perform one tableau transformation with the pivot element a_{rs} by using the Gauss-Jordan pivoting method described as Procedure A.19. The pivoting rules are applied to all elements of the tableau. Go to Step 1.

Note that the pivot column selection rule in Step 2 of this algorithm allows the user a large degree of freedom. Frequently the column with the smallest c_j (the "most negative" or "steepest unit ascent" rule) is chosen as pivot column since it results in the largest increase of the value of the objective function per unit increase of the entering variable; this is the rule applied throughout this book. A

large number of alternative rules exist, a good collection is found in Eiselt *et al.* (1987). One popular alternative is the "greatest change method," which determines pivot elements for each pivot-eligible column (i.e., each nonbasic column j with a negative indicator) and chooses the pivot that results in the largest increase of the value of the objective function. Extensive test have revealed that some pivot column selection rules result in savings in terms of iterations; such savings are, however, achieved at the cost of additional computations required to apply the rule. Overall, it does not appear to be worthwhile to invest too much time in different rules, especially as the information at any extreme point is only local at the present extreme point, while the optimality of a solution depends on local criteria at the optimal point, something now known during intermediate calculations.

As an illustration of the primal simplex method, consider the following numerical

Example (Phase 2 of the Simplex Method): Consider the following linear programming problem

$$
\begin{array}{lrll}
\text{P: Max } z = 40x_1 + 50x_2 + 100 & & & \\
\text{s.t.} & 2x_1 + x_2 & \le 12 & (I) \\
& -4x_1 + 5x_2 & \le 20 & (II) \\
& x_1 + 3x_2 & \le 15 & (III) \\
& x_1, \quad x_2 & \ge 0. &
\end{array}
$$

After the addition of slack variables S_1, S_2 and S_3, the initial tableau can be written as

T^1:

basis	x_1	x_2	S_1	S_2	S_3	1
S_1	2	1	1	0	0	12
S_2	−4	(5)	0	1	0	20
S_3	1	3	0	0	1	15
	−40	−50	0	0	0	100
		↑				

In tableau T^1, the basis consists of variable S_1 (the basic variable in the first row) whose present value equals 12, S_2 (in second row) with value 20, and S_3 (in third row) with value 15. Currently, x_1 and x_2 are nonbasic variables with values of zero. For this feasible solution, the value of the objective function is $z = 100$. Since both nonbasic variables have a negative entry in the objective function, either one could be chosen for introduction into the basis; here, we follow the "most negative" criterion and choose x_2. The pivot row is found by determination of $\min\left\{12/1; 20/5; 15/3\right\} = 20/5 = 4$, hence the second row is the pivot row with the pivot element $a_{22} = 5$, which is shown in the tableau. Thus the variable x_2 will enter the basis, while S_2 will leave the basis. Now the tableau is transformed and we obtain

T^2:

basis	x_1	x_2	S_1	S_2	S_3	1
S_1	$\frac{14}{5}$	0	1	$-\frac{1}{5}$	0	8
x_2	$-\frac{4}{5}$	1	0	$\frac{1}{5}$	0	4
S_3	$\frac{17}{5}$	0	0	$-\frac{3}{5}$	1	3
	−80	0	0	10	0	300

↑

with basic variables $S_1 = 8$, $x_2 = 4$, $S_3 = 3$ and nonbasic variables $x_1 = S_2 = 0$ and a value of the objective function of $z = 300$.

In tableau T^2, the only eligible pivot column is the x_1 column. The third row is the pivot row since $\frac{b_3}{a_{31}} = \min\left\{\frac{b_1}{a_{11}}, \frac{b_3}{a_{31}}\right\} = \min\left\{\frac{8}{\frac{14}{5}}; \frac{3}{\frac{17}{5}}\right\} = \frac{15}{17}$ (the second row is not pivot eligible, as it has a negative entry in the pivot column), i.e. the pivot element is a_{31}, meaning that x_1 enters and S_3 leaves the basis. After one tableau transformation we obtain the tableau T^3 with basic variables $S_1 = \frac{94}{17}$, $x_2 = \frac{80}{17}$, $x_1 = \frac{15}{17}$, nonbasic variables $S_2 = S_3 = 0$ and a value of the objective function $z = \frac{6{,}300}{17}$.

T^3:

basis	x_1	x_2	S_1	S_2	S_3	1
S_1	0	0	1	$\frac{5}{17}$	$-\frac{14}{17}$	$\frac{94}{17}$
x_2	0	1	0	$\frac{1}{17}$	$\frac{4}{17}$	$\frac{80}{17}$
x_1	1	0	0	$-\frac{3}{17}$	$\frac{5}{17}$	$\frac{15}{17}$
	0	0	0	$\frac{70}{17}$	$\frac{400}{17}$	$\frac{6{,}300}{17}$

Since S_2 has a negative element in the objective function, the solution in T^3 is still not optimal. The S_2 column is now chosen as pivot column; the first row is the pivot row since $\frac{b_1}{a_{14}} = \min\left\{\frac{\frac{94}{17}}{\frac{5}{17}}, \frac{\frac{80}{17}}{\frac{1}{17}}\right\} = \frac{94}{5}$ and a_{14} is the pivot, indicating that S_2 enters and S_1 leaves the basis. After one tableau transformation we obtain

T^4:

basis	x_1	x_2	S_1	S_2	S_3	1
S_2	0	0	$\frac{17}{5}$	1	$-\frac{14}{5}$	$\frac{94}{5}$
x_2	0	1	$-\frac{1}{5}$	0	$\frac{2}{5}$	$\frac{18}{5}$
x_1	1	0	$\frac{3}{5}$	0	$-\frac{1}{5}$	$\frac{21}{5}$
	0	0	14	0	12	448

Now all indicators in the objective function are nonnegative, thus the current basis is $\mathbf{x_B} = (S_2, x_2, x_1)$ and the corresponding solution $\bar{x}_1 = \frac{21}{5}$, $\bar{x}_2 = \frac{18}{5}$, $\bar{S}_1 = 0$, $\bar{S}_2 = \frac{94}{5}$, $\bar{S}_3 = 0$ is optimal with the associated value of the objective function $\bar{z} = 448$.

From now on we will delete the "basis" column in tableaus to simplify matters. This information can be recovered by identifying the basic columns in the tableau (they are the columns that are unit vectors) and the variables they are associated with. We should also point out that the basic matrix **B** consists of the original coefficients in the S_2, x_2 and x_1 columns, i.e.,

$$\mathbf{B} = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 5 & -4 \\ 0 & 3 & 1 \end{bmatrix}$$

and its inverse is found under the variables that were in the basis in the initial tableau, *viz*., (S_1, S_2, S_3), so that

$$\mathbf{B}^{-1} = \begin{bmatrix} \frac{17}{5} & 1 & -\frac{14}{5} \\ -\frac{1}{5} & 0 & \frac{2}{5} \\ \frac{3}{5} & 0 & -\frac{1}{5} \end{bmatrix},$$

so that the optimal solution is $\bar{\mathbf{x}}_\mathbf{B} = [\bar{S}_2, \bar{x}_2, \bar{x}_1]^T$

$$= \mathbf{B}^{-1}\mathbf{b} = \begin{bmatrix} \frac{17}{5} & 1 & -\frac{14}{5} \\ -\frac{1}{5} & 0 & \frac{2}{5} \\ \frac{3}{5} & 0 & -\frac{1}{5} \end{bmatrix} \begin{bmatrix} 12 \\ 20 \\ 15 \end{bmatrix} = \begin{bmatrix} \frac{94}{5} \\ \frac{18}{5} \\ \frac{21}{5} \end{bmatrix}$$

as indicated in the optimal tableau T^4. The graphical representation of the above problem is shown in Figure 3.12:

In Figure 3.12, the simplex procedure starts at point *O* (tableau T^1), moves to point *A* (tableau T^2), continues to point *B* (tableau T^3), and finally reaches point *C* (tableau T^4). Had we selected x_1 as entering variable in T^1, the simplex path would have been *O*, *D*, *C* instead.

At this point we would like to demonstrate how the changes of the basis can be observed in a graph. Note that the problem under consideration has $n = 2$ decision variables and $m = 3$ structural constraints, so that any basis will include exactly 3 variables. In order to identify them, it is useful to note that all variables that assume a positive value at some point must be basic.

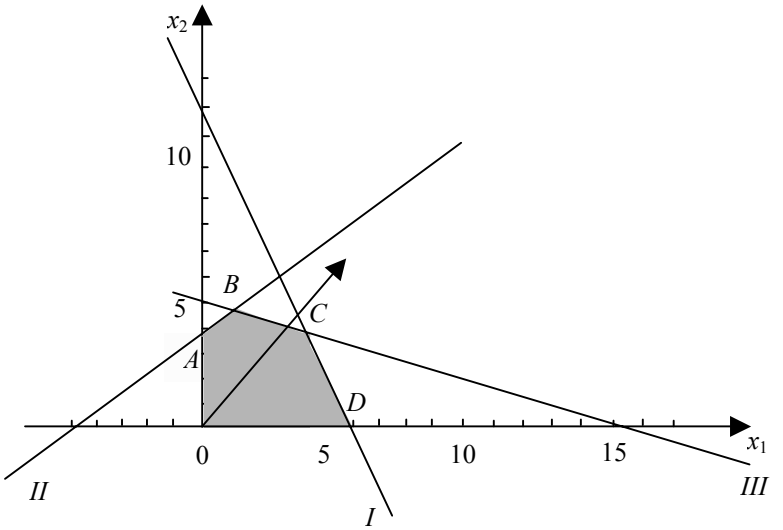


Figure 3.12

Start at the origin, at which the slack variables S_1, S_2, and S_3 are in the basis. The first move leads to point A. Here, x_2 is positive, so that it must be in the basis. (Observe that x_1 assumes a value of zero at point A, so that we cannot decide upon its status re: basic or nonbasic variable). Since the point A is located on hyperplane II, the slack variable S_2 will be zero, making it again impossible to decide its status. However, point A is not located on either of the other two hyperplanes, so that the slack variables S_1 and S_3 are positive, so that they must be basic variables. As a result, at point A, the variables x_2, S_1, and S_3 form the basis. As a result, the move from *0* to A was made by having x_2 enter the basis and having S_2 leave the basis.

Consider now the next move from point A to point B At point B, both decision variables x_1 and x_2 are positive, so that they are in the basis. Point B is not located on hyperplane I, so that the slack variable S_1 must be positive, so that it is the third basic variable. Thus the move from A to B constitutes a basis change in which S_1 enters the basis, while S_3 leaves the basis.

At the optimum point C, the basis consists of x_1, x_2, and S_2 (as point C is not located on hyperplane II, so that S_2 must be positive), meaning that the move from B to C was achieved by introducing S_2 into the basis and have S_1 leave the basis.

In the following we will drop the restriction $b_i > 0$ and $R_i = \{\leq\}$ and allow all relations of the type $\leq$, $=$ and $\geq$. The first preparatory step is to multiply all relations $\mathbf{a}_{i\bullet}\mathbf{x}\ R_i\ b_i$ with $b_i < 0$ by some negative number, leading to constraints that all have nonnegative right-hand side values. At this point we will add slack

variables, subtract excess variables, and add artificial variables wherever needed, as shown in Chapter 1.

As a result, the initial basis includes all slack and artificial variables but none of the decision and excess variables. Suppose now that at some stage during the calculations there is at least one artificial variable $A_i > 0$. If A_i were added to an original equality, then the assumption that $A_i > 0$ and the fact that $\mathbf{a}_{i\bullet}\mathbf{x} + A_i = b_i$ is always satisfied implies $\mathbf{a}_{i\bullet}\mathbf{x} \leq b_i$ which violates the original constraint $\mathbf{a}_{i\bullet}\mathbf{x} = b_i$.

Consider now an inequality $\mathbf{a}_{i\bullet}\mathbf{x} \geq b_i$ which is transformed to $\mathbf{a}_{i\bullet}\mathbf{x} - E_i + A_i = b_i$. The columns under E_i and A_i are $-\mathbf{e}_i$ and $\mathbf{e}_i$, respectively, and thus linearly dependent which implies that they cannot both be in the basis at the same time. If, by assumption, $A_i > 0$, then it must be a basic variable and E_i must be nonbasic with a value of zero which implies that $\mathbf{a}_{i\bullet}\mathbf{x} < b_i$, again a violation of the original constraint.

Summarizing we can state that the solution is not feasible, as long as there exists at least one $A_i > 0$. The purpose of the first phase of the simplex method is to reduce the values of all artificial variables to zero by driving them out of the basis. Since the nonnegativity constraints apply to all variables including artificial variables, we have $A_i \geq 0 \ \forall \ i$, so that $\sum_i A_i$ achieves a value of zero only if all artificial variables assume a value of zero (which is the desired situation), whereas $\sum_i A_i > 0$ implies that at least $A_i > 0$ is left and the current solution is not yet feasible. In order to drive all artificial variables out of the basis, we will use the *artificial objective function* (aof) Min $w = \sum_i A_i$ throughout the first phase. Since all artificial variables are in the initial basis, they have to be expressed in terms of nonbasic variables. Suppose that the constraints have been reordered, such that the first k rows belong to former $\leq$ relations and the remaining $(m - k)$ rows belong to original equations and $\geq$ relations, in which artificial variables are needed.

Furthermore, assume that the vector of variables $\mathbf{x}$ includes all variables except for the artificial variables. Then $\mathbf{a}_{i\bullet}\mathbf{x} + A_i = b_i \ \forall \ i = k + 1,\ldots, m$, or, equivalently, $A_i = b_i - \mathbf{a}_{i\bullet}\mathbf{x} \ \forall \ i = k + 1,\ldots, m$, so that the artificial objective function can now be written as $\text{Min} \sum_{i=k+1}^{m} A_i = \text{Min} \sum_{i=k+1}^{m} b_i - \sum_{i=k+1}^{m} a_{i\bullet}x$. Let now the parameters w_j be defined as $w_j = -\sum_{i=k+1}^{m} a_{ij}$, $j = 1,\ldots, n$ and $w_0 = -\sum_{i=k+1}^{m} b_i$; then the initial tableau can be written as shown in Table 3.2.

Table 3.2

x	S	E	A	1	
A	**I**	**0**	**0**	**b**	original ≤ constraints
	0	**−I**	***I***		original ≥ constraints
		0			original equalities
−c	**0**			z_0	gof: given objective function
w	**0**	**e**	**0**	w_0	aof: artificial objective function

We can now formally state the

The Primal Simplex Algorithm: Phase 1

Step 1: Is $w_j \geq 0 \; \forall \; j$?
If yes: Go to Step 3.
If no: Go to Step 2.

Step 2: Select some $w_s < 0$; the s-th column is then the pivot column.
Select the pivot row r, so that $\frac{b_r}{a_{rs}} = \min_{i=1,\ldots,m} \left\{ \frac{b_i}{a_{is} : a_{is} > 0} \right\}$. The element $a_{rs} > 0$ is the pivot. Perform a tableau transformation, using the Gauss-Jordan pivoting method described as Procedure A.19 Go to Step 1.

Step 3: Are all artificial variables nonbasic?
If yes: Drop the artificial objective function as well as all artificial variables and their columns from the tableau and go to Phase 2 of the primal simplex algorithm.
If no: Go to Step 4.

Step 4: Is $w_0 = 0$?
If yes: The current solution is feasible. Select any pivot $a_{rs} \neq 0$, so that $b_r = 0$ and some artificial variable A_i is basic in row r. Perform a tableau transformation with the Gauss-Jordan pivoting technique. Repeat this procedure until all artificial variables are

nonbasic. (If at some point $b_r = 0$, A_s is basic in row r, and all elements in row r except $a_{rs} = 1$ are zero, drop row r and column s from the tableau.) Delete all artificial variables and their columns and the artificial objective function from the tableau and go to Phase 2 of the primal simplex algorithm.

If no: Stop, the problem has no feasible solution.

In the above algorithm it is possible to drop an artificial variable and its column from a tableau as soon as it becomes nonbasic. This column drop rule should not be used for artificial variables associated with equation constraints, if postoptimality analyses (see Chapter 6 in this volume) are to be performed. Phases 1 and 2 are usually bundled and referred to collectively as the *Two Phase Method*.

We will now illustrate the two-phase method by means of the following

Example (The Two-Phase Simplex Method): Consider the following linear programming problem

$$
\begin{aligned}
&\text{P: Max } z = 3x_1 + x_2 \\
&\text{s.t.} \quad 3x_1 + 2x_2 \le 24 \qquad (I) \\
&\qquad\quad 4x_1 - x_2 \ge 8 \qquad (II) \\
&\qquad\quad x_1 - 2x_2 = 0 \qquad (III) \\
&\qquad\quad x_1, \; x_2 \ge 0.
\end{aligned}
$$

Adding a slack S_1 to the first constraint, subtracting an excess variable E_2 and adding an artificial variable A_2 to the second constraint, and adding an artificial variable A_3 to the third constraint, we obtain the initial tableau T^1.

Again, the artificial objective function coefficients are obtained by adding the coefficients in the columns in all rows, in which artificial variables are in the basis. In this example, these are the second and third rows. The meaning of this procedure becomes apparent when we rewrite the artificial variables in terms of nonbasic variables as $A_2 = 8 - 4x_1 + x_2 + E_2$ and $A_3 = -x_1 + 2x_2$, so that the artificial objective function Min $w = A_2 + A_3$ or Min $w - 8 = -5x_1 + 3x_2 + E_2$ which is exactly the artificial objective function in the above tableau.

T^1:	x_1	x_2	S_1	E_2	A_2	A_3	1
	3	2	1	0	0	0	24
	4	−1	0	−1	1	0	8
	(1)	−2	0	0	0	1	0
gof	−3	−1	0	0	0	0	0
aof	−5	3	0	1	0	0	−8
	↑						

Since w_1 is the only negative entry in the artificial objective function, the x_1 column must be selected as pivot column and is thus the entering variable (the fact that c_1 and $c_2 < 0$ does not matter at this stage). The minimum ratio rule then determines a_{31} as the pivot. Since x_1 enters the basis in the third row, A_3 leaves the basis and is immediately deleted. After one iteration, the new tableau is

T^2:	x_1	x_2	S_1	E_2	A_2	1
	0	8	1	0	0	24
	0	(7)	0	−1	1	8
	1	−2	0	0	0	0
gof	0	−7	0	0	0	0
aof	0	−7	0	1	0	−8
		↑				

Now w_2 is the only negative entry in the artificial objective function, and thus x_2 is introduced into the basis with a_{22} as pivot. This means that A_2 leaves the basis and at this point, no more artificial variables are in the basis; Phase 1 has now been terminated successfully (i.e., with a feasible solution), and the artificial objective function can be dropped form the tableau. (If we had kept the A_2 and A_3 columns as well as the artificial objective function, all coefficients in the artificial objective function would be zero except those under A_2 and A_3 which would be one.)

T^3:	x_1	x_2	S_1	E_2	1
	0	0	1	$(\frac{8}{7})$	$\frac{104}{7}$
	0	1	0	$-\frac{1}{7}$	$\frac{8}{7}$
	1	0	0	$-\frac{2}{7}$	$\frac{16}{7}$
gof	0	0	0	−1	8
				↑	

T^3 is also the first tableau in Phase 2 and the optimization proceeds with the usual optimality test that investigates the signs of the indicators in the given objective function row. Since $c_4 = -1 < 0$, the solution in T^3 is not yet optimal and the variable E_2 has to enter the basis. The variable enters the basis in the first row, so that the slack variable S_1 will leave the basis. After one more iteration we obtain tableau T^4 which is optimal with $\bar{\mathbf{x}} = (6, 3)$, $\bar{S}_1 = 0$, $\bar{E}_2 = 13$, and $\bar{z} = 21$.

T^4:	x_1	x_2	S_1	E_2	1
	0	0	$\frac{8}{7}$	1	13
	0	1	$\frac{1}{8}$	0	3
	1	0	$\frac{1}{4}$	0	6
gof	0	0	$\frac{7}{8}$	0	21

Notice that in the solution process in Phase 1 it is not necessarily the case that an artificial variable leaves the basis in each iteration. It is also worth mentioning that an artificial variable that is a nonbasic variable but has been kept in the tableau for some reason, should never be introduced into the basis, even if it has a negative indicator. Furthermore, a tableau can be optimal even if it has negative indicators under artificial variables. Steps 1 and 2 of the primal simplex algorithm have to be amended accordingly.

Figure 3.13 is a graphical representation of the above numerical example and it allows us to follow the simplex procedure. All feasible points are located on the line segment between points A and B. The procedure commences at point O (tableau T^1) with an infeasible solution, changes the basis but stays at point O (tableau T^2). This phenomenon is called primal degeneracy and is discussed in more detail below. The method then moves on to point A (tableau T^3) which terminates Phase 1, as point A is the first feasible solution encountered along the path. From there the method moves to the optimal solution at point B (tableau T^4).

In the literature and particularly in textbooks we frequently find the so-called *Big M Method* instead of the Two Phase Method. While their appearance is different, these two approaches are identical in principle. Rather than having a given objective function **cx** and an artificial objective function **wx** + **eE**, the Big M Method combines these two into one **cx** + **MA** = **cx** + **Mwx** + **MeE** with $M >> 0$ some unspecified, but sufficiently large, number. As an example, in tableau T^1 above, we have the two objective rows

$$\begin{bmatrix} -3, & -1, & 0, & 0, & 0, & 0; & 0 \\ -5, & 3, & 0, & 1, & 0, & 0; & -8 \end{bmatrix},$$

whereas the Big M Method combines them into the objective row $[-3 - 5M, -1 + 3M, 0, M, 0, 0; -8M]$. Since M is assumed to be sufficiently large, it dominates all other numbers and hence the pivot selection is identical to the one in the two phase method. We mention the Big M method here not because of its theoretical or computational merit. Instead, we believe that its philosophical background is rather interesting. While the two-phase method prohibits violations of feasibility, the Big M method allows them, but penalizes them by assigning a very high penalty to them. In particular, the Big M method as applied to a maximization problem will assign large negative objective function coefficients to all those variables which are to be driven out of the basis. In other words, every feasible solution that includes some $A_i > 0$ will have an unacceptably small value of the objective function. Penalty methods are quite applied to many different mathematical programming techniques, most frequently in the context of nonlinear programming.

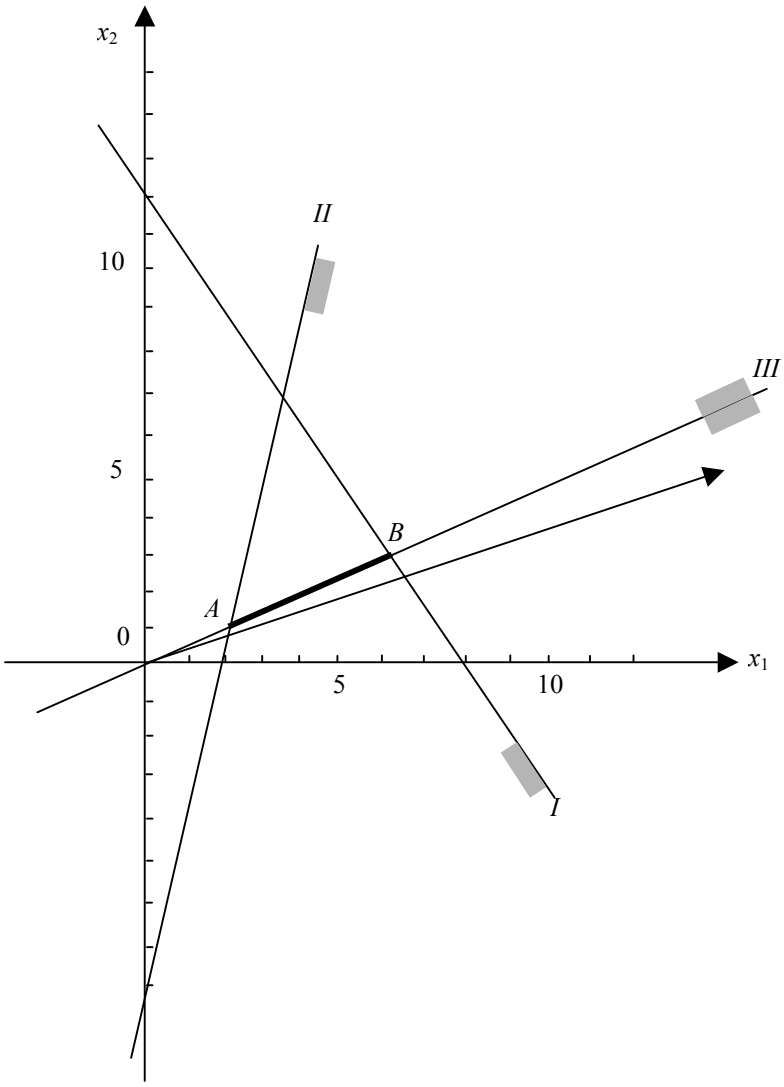


Figure 3.13

3.2.2 Four Special Cases Revisited

The remainder of this Chapter is dedicated to the discussion of the four special cases in linear programming, how they are recognized in the simplex tableau, and how to deal with them. In order to make transparent the relations between the graphical and algebraic representations of the problems, we use the same numerical examples here that we have used in the previous section when discussing the four special cases in the graphical context.

Case 1: No feasible solutions exist, if $w_0 < 0$ but $w_j \geq 0 \ \forall \ j$, i.e., at least one artificial variable $A_i > 0$ but no pivot column in the artificial objective function can be found.

Example: Consider the following linear programming problem example

$$\begin{aligned} \text{P: Max } z &= x_1 + x_2 \\ \text{s.t.} \quad x_1 &\leq 2 \\ x_2 &\leq 1 \\ 2x_1 + 5x_2 &\geq 10 \\ x_1, \ x_2 &\geq 0. \end{aligned}$$

The simplex tableaus for this problem are:

T^1:

	x_1	x_2	S_1	S_2	E_3	A_3	1
	1	0	1	0	0	0	2
	0	(1)	0	1	0	0	1
	2	5	0	0	−1	1	10
gof	−1	−1	0	0	0	0	0
aof	−2	−5	0	0	1	0	−10
		↑					

T^2:

	x_1	x_2	S_1	S_2	E_3	A_3	1
	(1)	0	1	0	0	0	2
	0	1	0	1	0	0	1
	2	0	0	−5	−1	1	5
gof	−1	0	0	1	0	0	1
aof	−2	0	0	5	1	0	−5
	↑						

T^3:

	x_1	x_2	S_1	S_2	E_3	A_3	1
	1	0	1	0	0	0	2
	0	1	0	1	0	0	1
	0	0	−2	−5	−1	1	1
gof	0	0	1	1	0	0	3
aof	0	0	2	5	1	0	−1

In tableau T^3, no pivot column can be found with respect to the artificial objective function, but $A_3 = 1 > 0$, hence no feasible solutions exist.

Case 2: Unbounded optimal solutions are encountered if there exists at least one j, such that $c_j < 0$ (i.e., a pivot-eligible column), and $a_{ij} \leq 0$, $i = 1,\ldots, m$.

Example: Consider the linear programming problem

P: Max $z = x_1 - 2x_2$
s.t. $-2x_1 + x_2 \le 2$
$x_1 - 3x_2 \le 3$
$x_1, \quad x_2 \ge 0$

The tableaus, generated during the simplex procedure, are:

T^1:

x_1	x_2	S_1	S_2	1
−2	1	1	0	2
1	−3	0	1	3
−1	2	0	0	0

↑

T^2:

x_1	x_2	S_1	S_2	1
0	−5	1	2	8
1	−3	0	1	3
0	−1	0	1	3

↑

In tableau T^2, the x_2 column is pivot eligible due to $c_2 < 0$ but no positive pivot can be found, hence unbounded optimal solutions exist. Even if many columns can be selected as pivot columns (all with $c_j < 0$), if only one among those does not include a possible pivot, i.e., a positive component, one can immediately abort the procedure with the conclusion that unbounded optimal solutions exist.

Case 3: Dual degeneracy occurs if a nonbasic variable has a zero coefficient in the objective function row. It indicates that the present solution has a neighboring extreme point that has the same value of the objective function as the current solution. If the current solution is optimal, then dual degeneracy indicates the existence of alternative optimal solutions. These adjacent solutions with identical values of the objective function are generated by selecting the nonbasic variable x_s for which $c_s = 0$ as entering variable and performing a regular tableau transformation.

Example 1: Consider the linear programming problem

P: Max $z = x_1 + x_2$
s.t. $x_1 + x_2 \ge 1$
$x_1 + x_2 \le 2$
$x_1, \quad x_2 \ge 0.$

The sequence of tableaus, leading to the solution, is:

T^1:

	x_1	x_2	E_1	A_1	S_2	1
	1	1	−1	1	0	1
	1	1	0	0	1	2
gof	−1	−1	0	0	0	0
aof	−1	−1	1	0	0	−1

↑

T^2:

	x_1	x_2	E_1	S_2	1
	1	1	−1	0	1
	0	0	1	1	1
gof	0	0	−1	0	1

↑

T^3:	x_1	x_2	E_1	S_2	1
	1	1	0	1	2
	0	0	1	1	1
gof	0	0	0	1	2

Note that T^2 marks the end of Phase 1; in T^2 the variables x_1 and S_2 are basic and x_2 is nonbasic although its column is identical to the one of x_1, but x_1 was introduced into the basis in the previous step. Since x_2 is nonbasic and $c_2 = 0$, an alternative solution to the one in T^2, where $\mathbf{x} = [1, 0]^T$ and $z = 1$, is generated by introducing x_2 into the basis; a_{12} is the only eligible pivot and the next tableau is identical to T^2 except the fact that the basis now includes x_2 and S_2, so that $\mathbf{x} = [0, 1]^T$ and $z = 1$. The same operation is possible in the optimal tableau T^3 where the basis includes x_1 and E_1 and $\bar{\mathbf{x}} = [2, 0]^T$ with $\bar{z} = 2$; again $c_2 = 0$ and x_2 could be introduced into the basis leading to the alternative optimal solution $\bar{\mathbf{x}} = [0, 2]^T$ with $\bar{z} = 2$.

Example 2: Consider the linear programming problem

$$
\begin{aligned}
\text{P: Max } z &= x_1 + x_2 + 2x_3 \\
\text{s.t.} \quad x_1 &\le 3 \\
x_2 &\le 5 \\
x_3 &\le 2 \\
x_1 + x_2 + 2x_3 &\le 10 \\
x_1,\ x_2,\ x_3 &\ge 0.
\end{aligned}
$$

This formulation has the same constraints as the example of Figure 3.5, but has a different objective function. One of the optimal bases of the problem is:

T^1:	x_1	x_2	x_3	S_1	S_2	S_3	S_4	1
	1	0	0	1	0	0	0	3
	0	0	0	1	1	2	−1	2
	0	0	1	0	0	1	0	2
	0	1	0	−1	0	−2	1	3
	0	0	0	0	0	0	1	10

The above tableau is clearly optimal with $\bar{\mathbf{x}}^1 = [3, 3, 2]^T$ and $\bar{z} = 10$, but the zero elements in the objective function under the nonbasic variables S_1 and S_3 indicate the existence of at least two other optimal solutions. In particular, the introduction of S_1 into the basis leads to T^2:

T^2:

x_1	x_2	x_3	S_1	S_2	S_3	S_4	1
1	0	0	0	−1	−2	1	1
0	0	0	1	1	2	−1	2
0	0	1	0	0	1	0	2
0	1	0	0	1	0	0	5
0	0	0	0	0	0	1	10

with the optimal solution $\bar{\mathbf{x}}^2 = [1, 5, 2]^T$ and $\bar{z} = 10$. Introducing the slack variable S_3 into the basis then leads to T^3:

T^3:

x_1	x_2	x_3	S_1	S_2	S_3	S_4	1
1	0	0	1	0	0	0	3
0	0	0	$\frac{1}{2}$	$\frac{1}{2}$	1	$-\frac{1}{2}$	1
0	0	1	$-\frac{1}{2}$	$-\frac{1}{2}$	0	$\frac{1}{2}$	1
0	1	0	0	1	0	0	5
0	0	0	0	0	0	1	10

with $\bar{\mathbf{x}}^3 = [3, 5, 1]^T$ and $\bar{z} = 10$. Note that $c_5 = 0$ and $c_6 = 0$ in T^2 again indicate alternative optimal solutions (introduction of S_2 into the basis leads back to T^1 whereas introduction of S_3 into the basis leads to T^3), and $c_4 = c_5 = 0$ in T^3 also indicate the existence of alternative optimal solutions (found in T^2 and T^1, respectively). Hence $\bar{\mathbf{x}}^1$, $\bar{\mathbf{x}}^2$ and $\bar{\mathbf{x}}^3$ are all optimal basic solutions. Now every linear convex combination of $\bar{\mathbf{x}}^1$, $\bar{\mathbf{x}}^2$ and $\bar{\mathbf{x}}^3$ is also optimal (but not a basic solution), i.e., $\bar{\mathbf{x}} = \lambda_1\bar{\mathbf{x}}^1 + \lambda_2\bar{\mathbf{x}}^2 + \lambda_3\bar{\mathbf{x}}^3$ is optimal for every $\lambda_1 + \lambda_2 + \lambda_3 = 1$ and $\lambda_1, \lambda_2, \lambda_3 \geq 0$. For instance, $\lambda = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$ generates the solution $\bar{\mathbf{x}} = \left[2\frac{1}{3}, 4\frac{1}{4}, 1\frac{2}{3}\right]^T$, while $\lambda = [\frac{1}{2}, \frac{1}{2}, 0]$ generates the solution $\bar{\mathbf{x}} = [2, 4, 2]^T$ and so forth, which are all optimal with $\bar{z} = 10$.

Example 3: An interesting situation is given if a tableau indicates unboundedness of optimal solutions and dual degeneracy simultaneously. Consider the linear programming problem

$$
\begin{aligned}
&\text{P: Max } z = -2x_1 + x_2 \\
&\text{s.t.} \quad -2x_1 + x_2 \leq 2 \\
&\qquad\quad x_1 - 3x_2 \leq 3 \\
&\qquad\quad x_1, \; x_2 \geq 0.
\end{aligned}
$$

The simplex tableaus are then

T^1:

x_1	x_2	S_1	S_2	1
−2	1	1	0	2
1	−3	0	1	3
2	−1	0	0	0

↑

T^2:

x_1	x_2	S_1	S_2	1
−2	1	1	0	2
−5	0	3	1	9
0	0	1	0	2

↑

indicating that the solution in T^2 is optimal but alternative optimal solutions exist due to $c_1 = 0$. If x_1 were to be increased by some $\varepsilon > 0$, the right-hand side would change to $\begin{bmatrix} 2+2\varepsilon \\ 9+5\varepsilon \end{bmatrix} = \begin{bmatrix} \bar{x}_1(\varepsilon) \\ \bar{S}_2(\varepsilon) \end{bmatrix}$. For instance, if $\varepsilon = 10$, we obtain $\bar{\mathbf{x}} = [10, 22]^T$ which is optimal with $\bar{z} = 2$ but not an extreme point. Dual degeneracy turns out to be a necessary, but not sufficient, condition for problems with unbounded optimal solutions that have a finite value of the objective function at optimum.

Case 4: Primal Degeneracy occurs, if one or more right-hand side values equal zero. If primal degeneracy is not built into the model right from the start (as in the example for the two-phase method in which primal degeneracy occurs at the origin) it is possible to anticipate its occurrence in the tableau that precedes primal degeneracy. In particular, if for any given pivot column, there are two or more rows that can alternatively be selected as pivot rows. Suppose that the *s*-th column has been selected as pivot column and the *r*-th and the *t*-th rows are tied for pivot row, i.e.,

$$\frac{b_r}{a_{rs}} = \frac{b_t}{a_{ts}} = \min_{i=1,\ldots,m} \left\{ \frac{b_i}{a_{is} : a_{is} > 0} \right\}.$$

Choosing a_{rs} as the pivot element, the *t*-th right-hand side in the next tableau will be $b_t := b_t - \frac{b_r a_{ts}}{a_{rs}} = b_t - \frac{b_r}{a_{rs}} a_{ts} = b_t - \frac{b_t}{a_{ts}} a_{ts} = 0$. A similar argument can be made if a_{ts} is chosen as the pivot. Primal degeneracy can occur at any basis, be it optimal, feasible, or infeasible. If primal degeneracy occurs at some stage during the iterations, it may remain in the tableau or it may vanish during the iterations that follow.

Example: Consider the linear programming problem

$$\begin{aligned} \text{P: Max } z = {} & x_1 + x_2 \\ \text{s.t.} \quad & 4x_1 + 2x_2 \le 9 \\ & 2x_1 + 3x_2 \le 6 \\ & 6x_1 + 5x_2 \le 15 \\ & x_1, \; x_2 \ge 0. \end{aligned}$$

The three tableaus, leading to the optimal solution of the problem, are shown below.

T^1:

x_1	x_2	S_1	S_2	S_3	1
(4)	2	1	0	0	9
2	3	0	1	0	6
6	5	0	0	1	15
−1	−1	0	0	0	0

↑

T^2:

x_1	x_2	S_1	S_2	S_3	1
1	$\frac{1}{2}$	$\frac{1}{4}$	0	0	$\frac{9}{4}$
0	(2)	$-\frac{1}{2}$	1	0	$\frac{3}{2}$
0	2	$-\frac{3}{2}$	0	1	$\frac{3}{2}$
0	$-\frac{1}{2}$	$\frac{1}{4}$	0	0	$\frac{9}{4}$

↑

T^3:

x_1	x_2	S_1	S_2	S_3	1
1	0	$\frac{3}{8}$	$-\frac{1}{4}$	0	$\frac{15}{8}$
0	1	$-\frac{1}{4}$	$\frac{1}{2}$	0	$\frac{3}{4}$
0	0	−1	−1	1	0
0	0	$\frac{1}{8}$	$\frac{1}{4}$	0	$\frac{21}{8}$

It is apparent that in tableau T^2, the elements a_{22} and a_{32} are tied for pivot since $\frac{b_2}{a_{22}} = \frac{b_3}{a_{32}} = \frac{3}{4}$, resulting in the optimal tableau T^3 that is primal degenerate.

The presence of primal degeneracy at any point of the computations may have serious computational consequences. As long as primal degeneracy is absent, $b_i > 0 \ \forall \ i$. With some pivot $a_{rs} > 0$, the new objective function value is $z := z - \frac{b_r c_s}{a_{rs}}$, so that, given $b_r > 0$ and $c_s < 0$, the objective function values are strictly increasing during the procedure. Since every basis is associated with exactly one z-value, it is not possible that a basis and/or a solution is repeated during the calculations. The fact that there is only a finite—albeit usually astronomically finite—number of bases proves the finiteness of the simplex procedure. This proof does, however, depend on the nonexistence of primal degeneracy.

Suppose now that primal degeneracy occurs at some stage during the iterations. Given that a row with a zero right-hand side value is chosen as a pivot row, we may generate a sequence of tableaus T^ν, $T^{\nu+1}$,…, T^μ with bases $\mathbf{x}_B^\nu$, $\mathbf{x}_B^{\nu+1}$,..., $\mathbf{x}_B^\mu$ with values of the objective function of $z^\nu = z^{\nu+1} = \ldots = z^\mu$. In other words, for a number of iterations the value of the objective function does not change. This is called *stalling*. Stalling may resolve itself if eventually a pivot in a nondegenerate row is chosen. However, it may happen that $\mathbf{x}_B^\nu = \mathbf{x}_B^\mu$, i.e., a basis, obtained somewhere in the process is again generated at a later stage, after which the procedure keeps repeating itself. Dantzig (1963) referred to this phenomenon as *circling*, but most

authors nowadays call it *cycling*. Cycling, if not avoided or properly dealt with, will prevent the simplex algorithm from being finite. The above discussion immediate leads to the following

Rule: Cycling can only occur in the presence of primal degeneracy; the presence of primal degeneracy, however, does not necessarily imply cycling.

The first occurrence of cycling was reported by Hoffman (1953). The smallest linear programming problem known to us in which cycling occurs was described by Beale (1955) and restated by Dantzig (1963). Its formulation is as follows:

$$
\begin{aligned}
\text{P: Max } z = & \tfrac{3}{4}x_1 - 150x_2 + \tfrac{1}{50}x_3 - 6x_4 \\
\text{s.t.} \quad & \tfrac{1}{4}x_1 - 60x_2 - \tfrac{1}{25}x_3 + 9x_4 \le 0 \\
& \tfrac{1}{2}x_1 - 90x_2 - \tfrac{1}{50}x_3 + 3x_4 \le 0 \\
& x_3 \le 1 \\
& x_1, \quad x_2, \quad x_3, \quad x_4 \ge 0.
\end{aligned}
$$

Given that the method always chooses the column with the most negative objective function coefficient as pivot column and the pivot row (if there are ties) as the highest row in the tableau, then tableau T^7 will include the same basis as T^1. For different pivot selection rules, primal degeneracy will still occur, but this will not necessarily result in cycling. One of the first techniques devised to prevent cycling was the *lexicographic selection rule* or *perturbation technique*. However, nowadays most authors suggest the use of Bland's (1977) rule, which is simple and straightforward to implement. Simply speaking, the rule chooses the entering variable as the pivot-eligible variable with the smallest subscript, and in case of a tie for the variable that leaves the basis, it selects the one with the smallest subscript as well. Another well-known simple rule is the LIFO (Last In, First Out) rule due to Zhang (1991). This rule selects as entering variable the pivot-eligible variable that most recently left the basis. A tie for the leaving variable is broken by choosing the variable that entered the basis most recently. For some further details on the subject, readers are referred to Dantzig and Thapa (1997).

Primal degeneracy occurs frequently in practice, e.g., in network flow problems, transportation problems, etc. It is of the utmost importance that commercial software codes for linear programming that are based on the simplex method properly treat degeneracy.

Chapter 4 DUALITY

Linear programming is based on the theory of duality. In essence, to each *primal* linear programming problem P, we can assign a *dual* linear programming problem P_D. This chapter formulates and discusses number of relations between the primal and dual problems that are not only essential to establish optimality conditions, but also provide insight into the problems and a meaningful economic interpretation of the optimization model. In particular, post optimality analyses depend almost entirely on the understanding of duality.

Far from being confined only to linear programming, duality theory has contributed considerably to the development of other mathematical optimization techniques. Intimately associated with Lagrangean functions and Lagrange multipliers, it has important applications in integer and nonlinear programming.

The origins of duality date back to John von Neumann (1947) and, independently, by Gale *et al*. (1951), followed by the contribution due to Dantzig and Orden (1953). The "optimale Geltungszahl" of Schmalenbach (1948) represents yet another independent development of the theory of duality in a managerial context. For a historical perspective, see Dantzig (1982).

This chapter is organized as follows. The first section presents all the necessary and preliminary results that lead to the statements and proofs of the weak and strong duality theorems as well as those of weak and strong complementary slackness. The next section considers primal-dual relations from a computational point of view in the context of the simplex method. The last section is devoted to economic interpretations of duality.

4.1 The Fundamental Theory of Duality

This section derives the fundamental theory of duality. We have attempted to rely as little as possible on references outside the domain of this book. The following exposition is a sequence of theorems and proofs, the first three of which belong to

a class called "Theorems of the Alternative." All these theorems include two related systems *I* and *II* of simultaneous linear equalities and/or inequalities, and the theorems state that either system *I* or system *II* has a solution, but never both. The proof is typically performed in two steps: first, we assume that system *I* has a solution and then show that system *II* cannot have a solution (in symbols: $I \rightarrow \overline{II}$ or "*I* implies not *II*"), and the second part assumes that system *I* has no solution and proves that system *II* must have a solution ($\bar{I} \rightarrow II$). Usually the first part is fairly simple whereas the second part may be rather lengthy. Mangasarian (1969) includes an extensive treatment of theorems of the alternative.

In the following we will assume that **A** is an $[m \times n]$-dimensional matrix with rank $rk\ \mathbf{A} = r$, **x** is an $[n \times 1]$-vector of variables, **b** is an $[m \times 1]$-vector of parameters, **u** is a $[1 \times m]$ vector of variables and ε is a scalar. Then we can state

Lemma 4.1 (Gale, 1960): Either the system *I*: **Ax= b** has a solution $\mathbf{x} \in \mathbb{R}^n$, or the system *II*: **uA = 0, ub** = ε ≠ 0 has a solution $\mathbf{u} \in \mathbb{R}^m$, but never both.

Proof: ($I \rightarrow \overline{II}$). Multiplication of **Ax = b** by **u** from the left and multiplication of **uA = 0** by **x** from the right results in **uAx = ub** and **uAx = 0**, respectively, hence **ub = 0** which contradicts that **u** solves system *II*.

($\bar{I} \rightarrow II$). System *II* can be written as $[\mathbf{A}, \mathbf{b}]^T\ \mathbf{u}^T = \begin{bmatrix} \mathbf{0} \\ \varepsilon \end{bmatrix}$, $\mathbf{u} \in \mathbb{R}^m$. Since, by assumption, system *I* has no solution, Theorem A.16 implies that $rk\ [\mathbf{A}, \mathbf{b}] > rk\ [\mathbf{A}]$ and since only column **b** is added to matrix **A** to form **[A, b]**, $rk\ [\mathbf{A}, \mathbf{b}] = rk\ [\mathbf{A}, \mathbf{b}]^T = r+1$. Hence $rk \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & \varepsilon \end{bmatrix}^T \geq r+1$. Since adding a zero vector to a matrix does not change its rank, $rk \begin{bmatrix} \mathbf{A} \\ \mathbf{0} \end{bmatrix}^T = r$ whereas the addition of any vector to a matrix can increase its rank by at most one, i.e., $rk \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & \varepsilon \end{bmatrix}^T \leq r+1$, so that $rk \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & \varepsilon \end{bmatrix}^T = r+1 = rk\ [\mathbf{A}, \mathbf{b}]^T$ and, according to Theorem A.16, system *II* must have at least one solution. □

Lemma 4.2 (Farkas' Lemma): Either the system *I*: **Ax = b**, **b ≠ 0** has a solution **x ≥ 0**, or the system *II*: **uA ≥ 0**, **ub < 0** has a solution $\mathbf{u} \in \mathbb{R}^m$, but never both.

Proof: ($I \rightarrow \overline{II}$)**:** Multiplication of **Ax= b** by **u** from the left and multiplication of **uA ≥ 0** by **x ≥ 0** from the right results in **uAx = ub** and **uAx ≥ 0**, respectively, so that **ub ≥** 0 which contradicts the second part of system *II***.**

$(\bar{I} \to II)$**.** *Case 1:* System *I* has no solution at all. Then Lemma 4.1 implies that $\mathbf{uA} = \mathbf{0}$, $\mathbf{ub} = \varepsilon \neq 0$ has a solution $\mathbf{u} \in \mathbb{R}^m$ and the selection of any $\varepsilon < 0$ results in a feasible solution for system *II*.

Case 2: System *I* has a solution $\mathbf{x} \in \mathbb{R}^n$ but not $\mathbf{x} \geq \mathbf{0}$. This portion of the proof will be carried out by induction on the columns of **A**. Consider $\mathbf{a}_{\bullet j}$, the *j*-th column of **A**, the variable x_j, and the right-hand side **b**. The fact that $\mathbf{a}_{\bullet j} x_j = \mathbf{b}$ has no solution $x_j \geq 0$ but a solution $x_j \in \mathbb{R}$ implies that $x_j < 0$. Division by x_j yields $\mathbf{a}_{\bullet j} = \dfrac{\mathbf{b}}{x_j}$ and in this case $\mathbf{u} = -\mathbf{b}^T$ is a solution to system *II* since $\mathbf{ua}_{\bullet j} = \dfrac{-\mathbf{b}^T\mathbf{b}}{x_j} > 0$ (because $\mathbf{b} \neq \mathbf{0}$ and $x_j < 0$) as well as $\mathbf{ub} = -\mathbf{b}^T\mathbf{b} < 0$.

Suppose now that the above result is true for $n - 1$ columns of **A**. By assumption, $\sum_{j=1}^{n} \mathbf{a}_{\bullet j} x_j = \mathbf{b}$ has no solution $x_j \geq 0$, $j = 1,\ldots, n$, which implies that $\sum_{j=1}^{n-1} \mathbf{a}_{\bullet j} x_j = \mathbf{b}$ has no solution $x_j \geq 0$, $j = 1,\ldots, n-1$**,** otherwise one could simply set $x_n = 0$. Also by assumption, there exists a vector $\bar{\mathbf{u}}$, such that $\bar{\mathbf{u}}\mathbf{a}_{\bullet j} \geq 0$ for $j = 1, \ldots, n-1$ and $\bar{\mathbf{u}}\mathbf{b} < 0$. Then either $\bar{\mathbf{u}}\mathbf{a}_{\bullet n} \geq 0$ in which case $\bar{\mathbf{u}}$ solves the system and the lemma is proved, or $\bar{\mathbf{u}}\mathbf{a}_{\bullet n} < 0$. In this case, define

$$\tilde{\mathbf{a}}_{\bullet j} := \mathbf{a}_{\bullet j} + \lambda_j \mathbf{a}_{\bullet n}, \text{ with } \lambda_j := -\frac{\bar{\mathbf{u}}\mathbf{a}_{\bullet j}}{\bar{\mathbf{u}}\mathbf{a}_{\bullet n}} \geq 0,\ j = 1, \ldots, n \text{ and}$$

$$\tilde{\mathbf{b}} := \mathbf{b} + \lambda_0 \mathbf{a}_{\bullet n}, \text{ with } \lambda_0 := -\frac{\bar{\mathbf{u}}\mathbf{b}}{\bar{\mathbf{u}}\mathbf{a}_{\bullet n}} < 0.$$

Then the system $\sum_{j=1}^{n-1} \tilde{a}_{\bullet j} \tilde{x}_j = \tilde{\mathbf{b}}$ can be rewritten as

$$\sum_{j=1}^{n-1} \mathbf{a}_{\bullet j} \tilde{x}_j + \mathbf{a}_{\bullet n} \left[\sum_{j=1}^{n-1} \lambda_j \tilde{x}_j - \lambda_0 \right] = \mathbf{b} \qquad (1)$$

This system has no solution $\tilde{x}_j \geq 0$, $j=1,\ldots, n-1$**,** because otherwise one could set $x_j := \tilde{x}_j \geq 0$**,** $j=1,\ldots, n-1$**,** and $x_n := \sum_{j=1}^{n-1} \lambda_j \tilde{x}_j - \lambda_0 > 0$ which contradicts the assumption that $\mathbf{Ax} = \mathbf{b}$ has no nonnegative solution **x**. If relation (1) has no solution $\tilde{x}_j \geq 0$, $j = 1,\ldots, n-1$, then—by assumption—there exists a vector $\tilde{\mathbf{u}}$,

such that $\tilde{\mathbf{u}}\tilde{\mathbf{a}}_{\bullet j} \geq 0$, $j = 1,..., n-1$ and $\tilde{\mathbf{u}}\tilde{\mathbf{b}} < \mathbf{0}$. Defining

$\mathbf{u}^* := \tilde{\mathbf{u}} - \dfrac{\tilde{\mathbf{u}}\mathbf{a}_{\bullet n}}{\bar{\mathbf{u}}\mathbf{a}_{\bullet n}}\bar{\mathbf{u}}$, we obtain

$$\mathbf{u}^*\mathbf{a}_{\bullet j} = \tilde{\mathbf{u}}\mathbf{a}_{\bullet j} - \frac{\tilde{\mathbf{u}}\mathbf{a}_{\bullet n}}{\bar{\mathbf{u}}\mathbf{a}_{\bullet n}}\bar{\mathbf{u}}\mathbf{a}_{\bullet j} = \tilde{\mathbf{u}}\mathbf{a}_{\bullet j} + \lambda_j\tilde{\mathbf{u}}\mathbf{a}_{\bullet n} = \tilde{\mathbf{u}}(\mathbf{a}_{\bullet j} + \lambda_j\mathbf{a}_{\bullet n}) = \tilde{\mathbf{u}}\tilde{\mathbf{a}}_{\bullet j} \geq 0$$

and

$\mathbf{u}^*\mathbf{a}_{\bullet n} = \tilde{\mathbf{u}}\mathbf{a}_{\bullet n} - \dfrac{\tilde{\mathbf{u}}\mathbf{a}_{\bullet n}}{\bar{\mathbf{u}}\mathbf{a}_{\bullet n}}\bar{\mathbf{u}}\mathbf{a}_{\bullet n} = 0$, which imply that $\mathbf{u}^*\mathbf{A} \geq \mathbf{0}$. Moreover

$$\mathbf{u}^*\mathbf{b} = \tilde{\mathbf{u}}\mathbf{b} - \frac{\tilde{\mathbf{u}}\mathbf{a}_{\bullet n}}{\bar{\mathbf{u}}\mathbf{a}_{\bullet n}}\bar{\mathbf{u}}\mathbf{b} = \tilde{\mathbf{u}}\mathbf{b} + \lambda_0\tilde{\mathbf{u}}\mathbf{a}_{\bullet n} = \tilde{\mathbf{u}}(\mathbf{b} + \lambda_0\mathbf{a}_{\bullet n}) = \tilde{\mathbf{u}}\tilde{\mathbf{b}} < \mathbf{0},$$

hence $\mathbf{u}^*\mathbf{b} < \mathbf{0}$ which implies that $\mathbf{u}^*$ is a solution to system *II*. This proves the lemma. □

Corollary 4.3 (Minkowski-Farkas lemma): Either the system *I*: $\mathbf{Ax} \leq \mathbf{b}, \mathbf{b} \neq \mathbf{0}$ has a solution $\mathbf{x} \geq \mathbf{0}$, or the system *II*: $\mathbf{uA} \geq \mathbf{0}, \mathbf{ub} < \mathbf{0}$ has a solution $\mathbf{u} \geq \mathbf{0}$, but never both.

Proof: Defining a matrix $\mathbf{B} = [\mathbf{A}, \mathbf{I}]$ and a vector of variables $\mathbf{z} = [\mathbf{x}, \mathbf{y}]$, Lemma 4.2 states that either *I*: $\mathbf{Bz} = \mathbf{b}, \mathbf{b} \neq \mathbf{0}$ has a solution $\mathbf{z} \geq \mathbf{0}$ or *II*: $\mathbf{uB} \geq \mathbf{0}, \mathbf{ub} < 0$ has a solution $\mathbf{u} \in \mathbb{R}^m$. This can be rewritten as *I*: **Ax + Iy = b, b ≠ 0** and **x, y ≥ 0** or, since **y ≥ 0**, as **Ax ≤ b**, **b ≠ 0, x ≥ 0** and *II*: **uA ≥ 0**, **uI ≥ 0**, **ub < 0, u** $\in \mathbb{R}^m$ or simply **uA ≥ 0**, **ub** < 0, **u ≥ 0.** □

Definition 4.4: A square matrix **D** = (d_{ij}) is said to be *antisymmetric* (or *skew symmetric*), if $d_{ij} = -d_{ji}\ \forall\ i, j$, i.e., $\mathbf{D} = -\mathbf{D}^T$.

Then we can prove the following

Lemma 4.5: If **D** is an antisymmetric matrix, then the system **Dx ≤ 0**, **x ≥ 0** has at least one solution $\bar{\mathbf{x}} \geq \mathbf{0}$, such that $\mathbf{D}\bar{\mathbf{x}} - \bar{\mathbf{x}} < \mathbf{0}$.

Proof: The proof is based on Corollary 4.3. Setting **A:= D** and **b:=** $\mathbf{e}_k$, Corollary 4.3 states that either system *I*: $\mathbf{Dx} \leq -\mathbf{e}_k$ has a solution $\mathbf{x} \geq \mathbf{0}$, or system *II*: **uD ≥ 0,** $-\mathbf{ue}_k = -\mathbf{u}_k < \mathbf{0}$ (or $\mathbf{u}_k > \mathbf{0}$) has a solution **u ≥ 0** but never both.

Case 1: Suppose that system *I* has a solution $\mathbf{x}^k$ and let x_j^k be the *j*-th component of that solution. Hence $\mathbf{Dx}^k \leq -\mathbf{e}_k$ and $\mathbf{x}^k \geq \mathbf{0}$ and, in particular, the *k*-th row is $\mathbf{d}_{k\bullet}\mathbf{x}^k \leq -1$; since $x_j^k \geq 0\ \forall\ j, \mathbf{d}_{k\bullet}\mathbf{x}^k - x_k^k \leq -1 < 0$ follows. For any other row $i \neq k$, $\mathbf{d}_{i\bullet}\mathbf{x}^k \leq 0$; since $x_j^k \geq 0\ \forall\ j, \mathbf{d}_{i\bullet}\mathbf{x}^k - \mathbf{x}_i^k \leq 0$ follows.

Case 2: Suppose that system *II* has a solution $\mathbf{u}^k$, then $\mathbf{u}^k\mathbf{D} \geq \mathbf{0}$, $u_k^k > 0$ and $\mathbf{u}^k \geq \mathbf{0}$. Note that $\mathbf{d}_{\bullet i} = -\mathbf{d}_{i\bullet}^T$, and set $\mathbf{x}^k := [\mathbf{u}^k]^T$. Since $\mathbf{u}^k\mathbf{d}_{\bullet i} \geq 0$ is satisfied, $-\mathbf{d}_{i\bullet}\mathbf{x}^k \geq 0$, or $\mathbf{d}_{i\bullet}\mathbf{x}^k \leq 0$ follows. By definition $x_k^k := u_k^k > 0$, and hence $\mathbf{d}_{k\bullet}\mathbf{x}^k - x_k^k < 0$ and $\mathbf{d}_{i\bullet}\mathbf{x}^k - \mathbf{x}_i^k \leq 0 \ \forall \ i \neq k$. Repeated application of the above procedure, i.e., $\mathbf{b} = -\mathbf{e}_k$, $k = 1,\ldots, n$ results in n solutions $\mathbf{x}^k$. Define

$$\bar{\mathbf{x}} := \sum_{k=1}^{n} \mathbf{x}^k, \text{ then } \mathbf{d}_{i\bullet}\bar{\mathbf{x}} - \bar{x}_i = \mathbf{d}_{i\bullet}\sum_{k=1}^{n} \mathbf{x}^k - \sum_{k=1}^{n} x_i^k =$$

$$= \underbrace{\left[\mathbf{d}_{i\bullet}\mathbf{x}^i - x_i^i\right]}_{<0} + \underbrace{\left[\mathbf{d}_{i\bullet}\sum_{k\neq i} \mathbf{x}^k - \sum_{k\neq i} \mathbf{x}_i^k\right]}_{\leq 0} < 0, i = 1,\ldots, n$$

which proves the lemma. □

Corollary 4.6: The system of simultaneous linear inequalities $\mathbf{Ax} - \lambda\mathbf{b} \leq \mathbf{0}$, $-\mathbf{uA} + \lambda\mathbf{c} \leq \mathbf{0}$, $\mathbf{ub} - \mathbf{cx} \leq 0$ with $\mathbf{x} \geq \mathbf{0}$, $\mathbf{u} \geq \mathbf{0}$, $\lambda \geq 0$ has at least one solution $\hat{\mathbf{x}}$, $\hat{\mathbf{u}}$, $\hat{\lambda}$, such that $\mathbf{A}\hat{\mathbf{x}} - \hat{\lambda}\mathbf{b} - \hat{\mathbf{u}}^{\mathbf{T}} < \mathbf{0}$, $-\hat{\mathbf{u}}\mathbf{A} + \hat{\lambda}\mathbf{c} - \hat{\mathbf{x}}^{\mathbf{T}} < \mathbf{0}$, $\hat{\mathbf{u}}\mathbf{b} - \mathbf{c}\hat{\mathbf{x}} - \hat{\lambda} < 0$.

Proof: In matrix notation, the above system can be written as $\mathbf{Dx} \leq \mathbf{0}$, $\mathbf{x} \geq \mathbf{0}$, where

$$\mathbf{D} := \begin{bmatrix} \mathbf{0} & \mathbf{A} & -\mathbf{b} \\ -\mathbf{A}^{\mathbf{T}} & \mathbf{0} & \mathbf{c}^{\mathbf{T}} \\ \mathbf{b}^{\mathbf{T}} & -\mathbf{c} & 0 \end{bmatrix} \text{ and } \mathbf{x} := \begin{bmatrix} \mathbf{u}^{\mathbf{T}} \\ \mathbf{x} \\ \lambda \end{bmatrix}$$

The application of Lemma 4.5 yields the desired result.□

In the following we will refer to the primal linear programming problem P and its associated dual P_D, defined as:

P: Max $z = \mathbf{cx}$			P_D: Min $z_D = \mathbf{ub}$
s.t. $\mathbf{Ax} \leq \mathbf{b}$		and	s.t. $\mathbf{uA} \geq \mathbf{c}$
$\mathbf{x} \geq \mathbf{0}$			$\mathbf{u} \geq \mathbf{0}$

The we can prove the following

Theorem 4.7 (The Weak Duality Theorem): If $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{u}}$ are feasible solutions for P and P_D, respectively, then $\mathbf{c}\tilde{\mathbf{x}} \leq \tilde{\mathbf{u}}\mathbf{b}$.

Proof: Since $\tilde{\mathbf{x}}$ is assumed to be feasible for P, $\mathbf{A}\tilde{\mathbf{x}} \le \mathbf{b}$ and multiplication by $\tilde{\mathbf{u}} \ge \mathbf{0}$ from the left results in $\tilde{\mathbf{u}}\mathbf{A}\tilde{\mathbf{x}} \le \tilde{\mathbf{u}}\mathbf{b}$, and since $\tilde{\mathbf{u}}$ is assumed to be feasible for P_D, $\tilde{\mathbf{u}}\mathbf{A} \ge \mathbf{c}$ and multiplication by $\tilde{\mathbf{x}}$ from the right results in $\tilde{\mathbf{u}}\mathbf{A}\tilde{\mathbf{x}} \ge \mathbf{c}\tilde{\mathbf{x}}$; hence $\mathbf{c}\tilde{\mathbf{x}} \le \tilde{\mathbf{u}}\mathbf{b}$. □

Now we are able to state the fundamental

Theorem 4.8 (The Strong Duality Theorem): Let a pair of primal and dual problems (P, P_D) be given. Then exactly one of the following three cases holds true:

(1) P and P_D have finite optimal solutions $\bar{\mathbf{x}}$ and $\bar{\mathbf{u}}$, respectively, such that $\mathbf{c}\bar{\mathbf{x}} = \bar{\mathbf{u}}\mathbf{b}$.
(2) Neither P nor P_D has a feasible solution.
(3) One of the problems has no feasible solution and the other has feasible, but no finite optimal solutions.

Proof: From Corollary 4.6, two distinct cases have to be considered, *viz*., $\hat{\lambda} > 0$ and $\hat{\lambda} = 0$.

Case 1: Suppose that $\hat{\lambda} > 0$. Define $\tilde{\mathbf{x}} := \frac{\hat{\mathbf{x}}}{\hat{\lambda}}$, $\tilde{\mathbf{u}} := \frac{\hat{\mathbf{u}}}{\hat{\lambda}}$ and $\tilde{\lambda} := \frac{\hat{\lambda}}{\hat{\lambda}} = 1$. According to Corollary 4.6, $\tilde{\mathbf{x}}, \tilde{\mathbf{u}}$ is a feasible solution to $\mathbf{A}\tilde{\mathbf{x}} \le \mathbf{b}$, $\tilde{\mathbf{u}}\mathbf{A} \ge \mathbf{c}$, and $\tilde{\mathbf{u}}\mathbf{b} \le \mathbf{c}\tilde{\mathbf{x}}$ with $\tilde{\mathbf{x}}, \tilde{\mathbf{u}} \ge \mathbf{0}$. Then $\mathbf{A}\tilde{\mathbf{x}} \le \mathbf{b}, \tilde{\mathbf{x}} \ge \mathbf{0}$ implies that $\tilde{\mathbf{x}}$ is also a feasible solution for the primal problem P, whereas $\tilde{\mathbf{u}}\mathbf{A} \ge \mathbf{c}$ and $\tilde{\mathbf{u}} \ge \mathbf{0}$ imply that $\tilde{\mathbf{u}}$ is a feasible solution for the dual problem $\mathbf{P}_D$, so that Theorem 4.7 applies and $\mathbf{c}\tilde{\mathbf{x}} \le \tilde{\mathbf{u}}\mathbf{b}$; hence $\mathbf{c}\tilde{\mathbf{x}} = \tilde{\mathbf{u}}\mathbf{b}$. Since $\mathbf{cx}$ is to be maximized but bounded by $\tilde{\mathbf{u}}\mathbf{b}$ from above and $\mathbf{ub}$ is to be minimized but bounded by $\mathbf{c}\tilde{\mathbf{x}}$ from below and case (1) in the theorem holds true, so that $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{u}}$ must be optimal solutions for the respective problems.

Case 2: $\hat{\lambda} = 0$. According to Corollary 4.6, the system $\mathbf{Ax} \le \mathbf{0}$, $\mathbf{uA} \ge \mathbf{0}$, and $\mathbf{ub} \le \mathbf{cx}$ with $\mathbf{x}, \mathbf{u} \ge \mathbf{0}$ has at least one solution $\hat{\mathbf{x}}, \hat{\mathbf{u}} \ge 0$ that satisfies $\mathbf{c}\hat{\mathbf{x}} > \hat{\mathbf{u}}\mathbf{b}$. Let $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{u}}$ be feasible solution for P and P_D respectively, i.e., $\mathbf{A}\tilde{\mathbf{x}} \le \mathbf{b}, \tilde{\mathbf{x}} \ge \mathbf{0}$ as well as $\tilde{\mathbf{u}}\mathbf{A} \ge \mathbf{c}, \tilde{\mathbf{u}} \ge \mathbf{0}$. Multiplication of $\mathbf{A}\tilde{\mathbf{x}} \le \mathbf{b}$ by $\hat{\mathbf{u}} \ge \mathbf{0}$ from the left results in $0 \le \hat{\mathbf{u}}\mathbf{A}\tilde{\mathbf{x}} \le \hat{\mathbf{u}}\mathbf{b}$, multiplication of $\tilde{\mathbf{u}}\mathbf{A} \ge \mathbf{c}$ by $\hat{\mathbf{x}} \ge \mathbf{0}$ from the right results in $0 \ge \tilde{\mathbf{u}}\mathbf{A}\hat{\mathbf{x}} \ge \mathbf{c}\hat{\mathbf{x}}$, hence $\mathbf{c}\hat{\mathbf{x}} \le \hat{\mathbf{u}}\mathbf{b}$ which contradicts $\mathbf{c}\hat{\mathbf{x}} > \hat{\mathbf{u}}\mathbf{b}$. Thus, in the case of $\hat{\lambda} = 0$, it is not possible for both problems P and P_D to have at least one feasible solution each and case (2) or (3) in the theorem holds true.

Suppose now that P has a feasible solution $\tilde{\mathbf{x}}$, i.e., $\mathbf{A}\tilde{\mathbf{x}} \le \mathbf{b}$ and $\tilde{\mathbf{x}} \ge \mathbf{0}$. Let $\hat{\mathbf{x}}$ be again one of the solutions of Corollary 4.6, then $\tilde{\mathbf{x}} + \varepsilon\hat{\mathbf{x}}, \varepsilon > \mathbf{0}$ is also a feasible solution for P, since $\tilde{\mathbf{x}}, \hat{\mathbf{x}} \ge \mathbf{0}$, the parameter $\varepsilon > 0$ implies that $\tilde{\mathbf{x}} + \varepsilon\hat{\mathbf{x}} \ge \mathbf{0}$, and $\mathbf{A}(\tilde{\mathbf{x}} + \varepsilon\hat{\mathbf{x}}) = \mathbf{A}\tilde{\mathbf{x}} + \varepsilon\mathbf{A}\hat{\mathbf{x}} \le \mathbf{b}$ as $\mathbf{A}\tilde{\mathbf{x}} \le \mathbf{b}$ and $\mathbf{A}\hat{\mathbf{x}} \le \mathbf{0}, \varepsilon > \mathbf{0}$. According to Corollary 4.6, $\hat{\mathbf{u}} \ge \mathbf{0}$ and $\hat{\mathbf{u}}\mathbf{A} \ge \mathbf{0}$, so that $\hat{\mathbf{u}}\mathbf{A}\tilde{\mathbf{x}} \ge \mathbf{0}$. On the other hand, feasibility of $\tilde{\mathbf{x}}$ for P implies $\mathbf{A}\tilde{\mathbf{x}} \le \mathbf{b}$ and thus $\hat{\mathbf{u}}\mathbf{A}\tilde{\mathbf{x}} \le \hat{\mathbf{u}}\mathbf{b}$. A combination of the results then yields $\hat{\mathbf{u}}\mathbf{b} \ge \mathbf{0}$. This, together with $\mathbf{c}\hat{\mathbf{x}} > \hat{\mathbf{u}}\mathbf{b}$ results in $\mathbf{c}\hat{\mathbf{x}} > \mathbf{0}$, so that the new solution $\tilde{\mathbf{x}} + \varepsilon\hat{\mathbf{x}}, \varepsilon > 0$ has a value of the objective function of P which is $\mathbf{c}(\tilde{\mathbf{x}} + \varepsilon\hat{\mathbf{x}}) = \mathbf{c}\tilde{\mathbf{x}} + \varepsilon\mathbf{c}\hat{\mathbf{x}}, \varepsilon > 0, \mathbf{c}\hat{\mathbf{x}} > 0$. Since ε can be increased arbitrarily, problem P has no finite optimum and case (3) in the theorem applies. A similar argument can be applied if P_D has at least one feasible solution. This proves the theorem. □

For additional reading, see Gale *et al.* (1951), Dantzig and Orden (1953), and Kuhn and Tucker (1956) who present some of the original developments of this material.

According to Theorem 4.8, a pair $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ of solutions for P and P_D respectively, is optimal if and only if $\mathbf{c}\bar{\mathbf{x}} = \bar{\mathbf{u}}\mathbf{b}$. An alternative optimality criterion is given in

Theorem 4.9 (Weak Complementary Slackness Theorem): If $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{u}}$ are feasible solutions for P and P_D, respectively, and satisfy

$$\bar{\mathbf{u}}(\mathbf{A}\bar{\mathbf{x}} - \mathbf{b}) = 0 \text{ and} \tag{2}$$

$$(\bar{\mathbf{u}}\mathbf{A} - \mathbf{c})\bar{\mathbf{x}} = 0 \tag{3}$$

then $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ is a pair of optimal solutions, and vice versa.

Proof: Owing to the strong duality theorem, it is sufficient to show that the above two conditions are equivalent to $\mathbf{c}\bar{\mathbf{x}} = \bar{\mathbf{u}}\mathbf{b}$. Rewriting them as $\bar{\mathbf{u}}\mathbf{A}\bar{\mathbf{x}} = \bar{\mathbf{u}}\mathbf{b}$ and $\bar{\mathbf{u}}\mathbf{A}\bar{\mathbf{x}} = \mathbf{c}\bar{\mathbf{x}}$, sufficiency follows. Then necessity condition can be demonstrated by multiplying $\mathbf{c}\bar{\mathbf{x}} = \bar{\mathbf{u}}\mathbf{b}$ by (−1) and adding $\bar{\mathbf{u}}\mathbf{A}\bar{\mathbf{x}}$ to both sides, resulting in $-\bar{\mathbf{u}}\mathbf{b} + \bar{\mathbf{u}}\mathbf{A}\bar{\mathbf{x}}$ or $\bar{\mathbf{u}}(\mathbf{A}\bar{\mathbf{x}} - \mathbf{b}) = (\bar{\mathbf{u}}\mathbf{A} - \mathbf{c})\bar{\mathbf{x}}$. Since both $\bar{\mathbf{x}}$ and $\bar{\mathbf{u}}$ are assumed to be feasible for their respective problems, $\bar{\mathbf{u}} \ge \mathbf{0}$ and $\mathbf{A}\bar{\mathbf{x}} - \mathbf{b} \le \mathbf{0}$ or $\bar{\mathbf{u}}(\mathbf{A}\bar{\mathbf{x}} - \mathbf{b}) \le 0$, whereas $(\bar{\mathbf{u}}\mathbf{A} - \mathbf{c}) \ge \mathbf{0}$ and $\bar{\mathbf{x}} \ge \mathbf{0}$, so that $(\bar{\mathbf{u}}\mathbf{A} - \mathbf{c})\bar{\mathbf{x}} \ge 0$. The above equality forces both terms to zero which yields the desired result. □

It should be pointed out that condition (2) in Theorem 4.9 can be written as $\sum_{i=1}^{m} \bar{u}_i(\mathbf{a}_{i\bullet}\bar{\mathbf{x}} - b_i) = 0$. Since $\bar{u}_i \ge 0$ and $\mathbf{a}_{i\bullet}\bar{\mathbf{x}} - b_i \le 0, i = 1,..., m$ due to primal

and dual feasibility, every single term of the sum has to equal zero. An equivalent argument can be applied to condition (3), so that (2) and (3) can be written as

$$\bar{u}_i(\mathbf{a}_{i\bullet}\bar{\mathbf{x}} - b_i) = 0,\ i = 1,\ldots, m \tag{2'}$$
$$(\bar{\mathbf{u}}\mathbf{a}_{\bullet j} - c_j)\bar{x}_j = 0,\ j = 1,\ldots, n \tag{3'}$$

An immediate consequence of conditions (2) and (3) is the following

Corollary 4.10: Satisfying conditions (2') and (3') is equivalent to

$$\bar{u}_i > 0 \quad \rightarrow\ \mathbf{a}_{i\bullet}\bar{\mathbf{x}} = b_i \tag{4}$$
$$\mathbf{a}_{i\bullet}\bar{\mathbf{x}} < b_i \ \rightarrow\ \bar{u}_i = 0 \tag{5}$$
$$\bar{x}_j > 0 \quad \rightarrow\ \bar{\mathbf{u}}\mathbf{a}_{\bullet j} = c_j \tag{6}$$
$$\bar{\mathbf{u}}\mathbf{a}_{\bullet j} > c_j \rightarrow\ \bar{x}_j = 0 \tag{7}$$

Finally, we will prove

Theorem 4.11 (Strong Complementary Slackness Theorem): If problems P and P_D have at least one feasible solution each, then there exists at least one pair of optimal solutions $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$, such that

$$\left(\mathbf{b} - \mathbf{A}\bar{\mathbf{x}}\right) + \bar{\mathbf{u}}^T > \mathbf{0} \tag{8}$$

and

$$\left(\bar{\mathbf{u}}\mathbf{A} - \mathbf{c}\right) + \bar{\mathbf{x}}^T > \mathbf{0}\,. \tag{9}$$

Proof: Since both problems are assumed to possess feasible solutions, we are dealing with case 1 in Theorem 4.8, where $\hat{\lambda} > 0$. Let $\hat{\mathbf{x}}, \hat{\mathbf{u}}$ and $\hat{\lambda}$ denote a solution to the system in Corollary 4.6, then $\mathbf{A}\hat{\mathbf{x}} - \hat{\lambda}\mathbf{b} - \hat{\mathbf{u}}^T < \mathbf{0}$ and $-\hat{\mathbf{u}}\mathbf{A} + \hat{\lambda}\mathbf{c} - \hat{\mathbf{x}}^T < \mathbf{0}$. Defining $\bar{\mathbf{x}} := \frac{\hat{\mathbf{x}}}{\hat{\lambda}}, \bar{\mathbf{u}} := \frac{\hat{\mathbf{u}}}{\hat{\lambda}}$, we obtain $\mathbf{A}\bar{\mathbf{x}} - \mathbf{b} - \bar{\mathbf{u}}^T < \mathbf{0}$ and $-\bar{\mathbf{u}}\mathbf{A} + \mathbf{c} - \bar{\mathbf{x}}^T < \mathbf{0}$ which is the desired result. □

Complementary slackness conditions are of major importance in primal-dual algorithms.

For a geometric representation of optimality, consider the following pair of primal and dual problems:

$$\begin{array}{lll} \text{P}: \text{Max } z = \mathbf{cx} & & \text{P}_\text{D}: \text{Min } z_\text{D} = \mathbf{ub} \\ \text{s.t.} \quad \mathbf{Ax} \le \mathbf{b} & \text{and} & \text{s.t.} \quad \mathbf{uA} \ge \mathbf{c} \\ \mathbf{x} \ge \mathbf{0} & & \mathbf{u} \ge \mathbf{0}. \end{array}$$

Furthermore, consider some extreme point $\bar{\mathbf{x}}$. It is now possible to demonstrate that $\bar{\mathbf{x}}$ is optimal only if the gradient of the objective function at $\bar{\mathbf{x}}$ can be represented as a nonnegative linear combination of the gradients of all hyperplanes which are binding at $\bar{\mathbf{x}}$. Formally, let $I := \{\mathbf{a}_{i\bullet}\bar{\mathbf{x}} = b_i\}$ denote the set of hyperplanes that are binding at $\bar{\mathbf{x}}$. Their respective gradients are $\mathbf{a}_{i\bullet}$ and a nonnegative linear combination is $\bar{\mathbf{u}}\mathbf{a}_{i\bullet}$, $i \in I$ with $\bar{\mathbf{u}} \ge \mathbf{0}$. Since the gradient of the objective function is $\mathbf{c}$, we set $u_i := 0 \; \forall \, i \notin I$ and the above condition reads $\sum_{i \in I} \bar{u}_i a_{ij} = c_j$, $j = 1, \ldots, n$. If this holds, then

$$\mathbf{c}\bar{\mathbf{x}} = \sum_{j=1}^{n} \sum_{i=1}^{m} c_j x_j = \sum_{j=1}^{n} \sum_{i \in I} \bar{u}_i a_{ij} \bar{x}_j + \sum_{j=1}^{n} \underbrace{\sum_{i \notin I} \bar{u}_i}_{=0} a_{ij} \bar{x}_j = \sum_{i \in I} \bar{u}_i \underbrace{\sum_{j=1}^{n} a_{ij} \bar{x}_j}_{=b_i} = \sum_{i=1}^{m} \bar{u}_i b_i = \bar{\mathbf{u}}\mathbf{b}$$

which must hold at optimum.

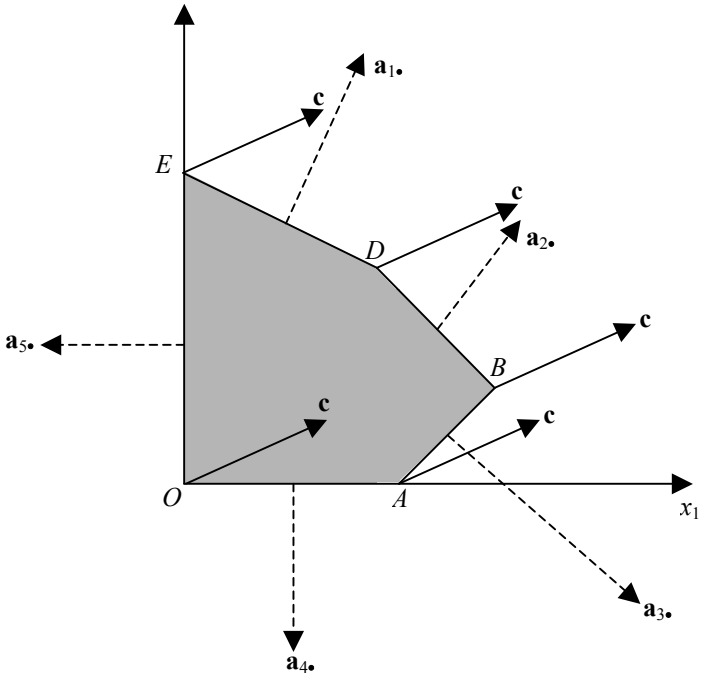


Figure 4.1a

A graphical representation of the above relation is shown in Figure 4.1a, where the gradient of the objective function is shown by solid lines at each extreme point, while the gradients of the constraints are displayed by broken lines.

Take, for instance, point *A*. The third and fourth hyperplanes are binding at *A*, but **c** can not be expressed as a nonnegative linear combination of $\mathbf{a}_3$, and $\mathbf{a}_{4\bullet}$, i.e. a vector pointed toward the cone given by $\mathbf{a}_{3\bullet}$ and $\mathbf{a}_{4\bullet}$. The same can be said for points *O*, *D* and *E*. Only for point *B*, there are values $\bar{u}_2, \bar{u}_3 \geq 0$ such that $\bar{u}_2\mathbf{a}_{2\bullet} + \bar{u}_3\mathbf{a}_{3\bullet} = \mathbf{c}$ and therefore *B* must be optimal.

Note also that since $\bar{u}_2, \bar{u}_3 > 0$ at point *B*, setting $\bar{u}_1 = \bar{u}_4 = \bar{u}_5 = 0$ the complementary slackness conditions are also fulfilled. However, this is not necessarily always the case. Take the same example but with a different objective function as shown in Figure 4.1b.

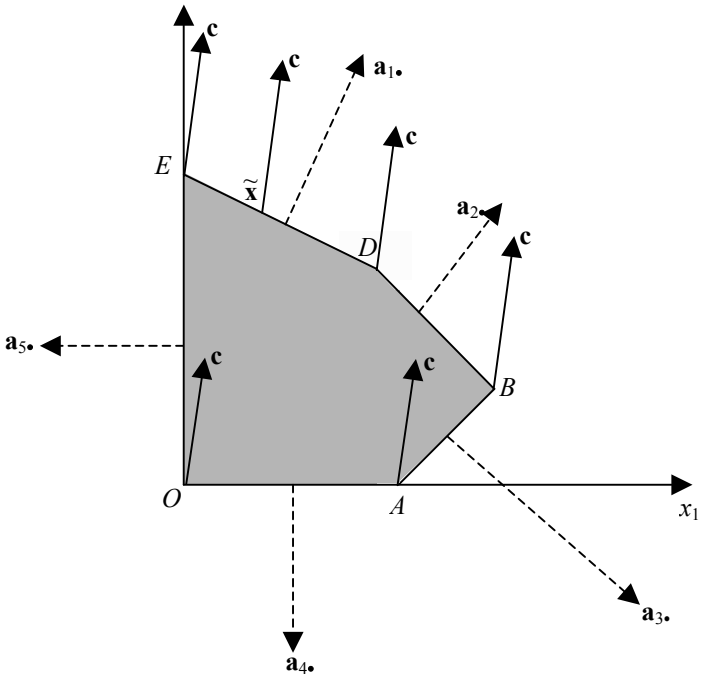


Figure 4.1b

Clearly, both points *D* and *E* are optimal solutions. Consider, for instance, point *D*. The first and second constraints are binding (and immediately $\bar{u}_3 = \bar{u}_4 = \bar{u}_5 = 0$) and the gradient of the objective function can be expressed as $\mathbf{c} = \bar{u}_1\mathbf{a}_{1\bullet} + \bar{u}_2\mathbf{a}_{2\bullet}$. However, $\bar{u}_2 = 0$ and the strong complementary slackness

condition $(b_i - \mathbf{a}_{i\bullet}\bar{\mathbf{x}}) + \bar{u}_i > 0$ is not satisfied for $i = 2$ because $b_2 - \mathbf{a}_{2\bullet}\bar{\mathbf{x}} = 0$ and $\bar{u}_2 = 0$. A similar line of reasoning can be applied to *E*. Again, the strong complementary slackness condition is violated.

Consider now some nonnegative linear combination of points *D* and *E*, such as the point $\tilde{\mathbf{x}}$, then $b_i - \mathbf{a}_{i\bullet}\tilde{\mathbf{x}} > 0$, $i = 2, 3, 4, 5$ and $b_1 - \mathbf{a}_{1\bullet}\tilde{\mathbf{x}} = 0$. Setting $\bar{u}_1 = 1$ and $\bar{u}_i = 0$, $i = 2, 3, 4, 5$ satisfies the strong complementary slackness condition and this is sufficient since it need only be satisfied for at least one optimal solution.

The remainder of this section will summarize and apply at least some of the theoretical results established above. According to the strong duality theorem, a pair of solutions $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ is optimal if and only if

(a) $\bar{\mathbf{x}}$ is feasible of P	(primal feasibility)
(b) $\bar{\mathbf{u}}$ is feasible of P_D	(dual feasibility)
(c) $\mathbf{c}\bar{\mathbf{x}} = \bar{\mathbf{u}}\mathbf{b}$	(complementary slackness)

Considering the following three systems P, P_D and P*:

$$\begin{array}{lll} \text{P: Max } z = \mathbf{cx} & \text{P}_D\text{: Min } z_D = \mathbf{ub} & \text{P*: } \mathbf{Ax} \le \mathbf{b} \\ \text{s.t. } \mathbf{Ax} \le \mathbf{b} & \text{s.t. } \mathbf{uA} \ge \mathbf{c} & \mathbf{x} \ge \mathbf{0} \\ \mathbf{x} \ge \mathbf{0} & \mathbf{u} \ge \mathbf{0} & \mathbf{uA} \ge \mathbf{c} \\ & & \mathbf{u} \ge \mathbf{0} \\ & & \mathbf{cx} - \mathbf{ub} = \mathbf{0}, \end{array}$$

then the following corollary is an immediate consequence of the results derived above.

Corollary 4.12: The following two statements are equivalent:

(*i*) $\bar{\mathbf{x}}$ is an optimal solution for P and $\bar{\mathbf{u}}$ is an optimal solution for P_D.
(*ii*) $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ is a feasible solution for P*.

In other words, rather than to optimize problem P or P_D, one could attempt to find feasible solutions for P*; a concept which is important for some non-simplex based solution techniques, a collection of which is described in Chapter 7 of this part of the book. It should also be mentioned that it is possible to derive the optimality conditions stated and proved above by way of Lagrangean functions and derivatives. For details, readers are referred to Eiselt *et al*. (1987).

Below we will describe specific rules, which may be used to set up the dual problem, associated with a given primal. To simplify matters, we assume without loss of generality that the primal is a maximization problem and there are only constraints of types ≤ and =. If either of the conditions were violated originally,

simple transformations as those described in Chapter 8 of this volume can help to reduce the problem to the desired form. The rules to establish a dual problem P_D associated with a given primal problem P are then summarized in Table 4.1.

Table 4.1

	Primal problem P (variables $x_j, j = 1,\ldots, n$)	*Dual problem* P_D (variables $u_i, i = 1,\ldots, m$)
(1)	$\mathbf{a_{i\bullet}x} \leq b_i$	$u_j \geq 0$
(2)	$\mathbf{a_{i\bullet}x} = b_i$	$u_j \in \mathbb{R}$
(3)	$x_j \geq 0$	$\mathbf{ua_{\bullet j}} \geq c_j$
(4)	$x_j \in \mathbb{R}$	$\mathbf{ua_{\bullet j}} = c_j$
(5)	Max $z = \mathbf{cx}$	Min $z_D = \mathbf{ub}$

The application of these rules can be illustrated by the following

Example: Consider the following linear programming problem:

$$
\begin{aligned}
\text{P: Min } z = {} & 3x_1 + x_2 \\
\text{s.t.} \quad & 3x_1 + 2x_2 \leq 24 \\
& 4x_1 - x_2 \geq 8 \\
& x_1 = 2x_2 \\
& x_1 \geq 0 \\
& x_2 \in \mathbb{R}.
\end{aligned}
$$

Transforming the problem into an equivalent problem that satisfies the above assumption results in

$$
\begin{aligned}
\text{P': Max } -z = {} & -3x_1 - x_2 \\
\text{s.t.} \quad & 3x_1 + 2x_2 \leq 24 \\
& -4x_1 + x_2 \leq -8 \\
& x_1 - 2x_2 = 0 \\
& x_1 \geq 0 \\
& x_2 \in \mathbb{R}.
\end{aligned}
$$

Assigning dual variables $u_1 \geq 0$ and $u_2 \geq 0$ to the first two primal constraints respectively (Rule 1) as well as a dual variable $u_3 \in \mathbb{R}$ to the third primal constraint (Rule 2), the dual is

$$
\begin{aligned}
\text{P}_\text{D}\text{: Min } -z_D = {} & 24u_1 - 8u_2 && \text{(Rule 5)} \\
\text{s.t.} \quad & 3u_1 - 4u_2 + u_3 \geq -3 && \text{(Rule 3)} \\
& 2u_1 + u_2 - 2u_3 = -1 && \text{(Rule 4)} \\
& u_1, \quad u_2 \geq 0 && \text{(Rule 1)} \\
& u_3 \in \mathbb{R} && \text{(Rule 2).}
\end{aligned}
$$

A more general illustration is provided in the following

Example: Consider the following linear programming problem

$$\begin{aligned} \text{P: Min } z &= \mathbf{cx} - \mathbf{dy} \\ \text{s.t.} \quad \mathbf{Ax} + \mathbf{By} &\le \mathbf{b}^1 \\ \mathbf{Dx} + \mathbf{Ey} &= \mathbf{b}^2 \\ \mathbf{Fx} + \mathbf{Gy} &\ge \mathbf{b}^3 \\ \mathbf{X} \qquad &\ge \mathbf{0} \\ \mathbf{y} &\in \mathbb{R}^r, \end{aligned}$$

where $\mathbf{x}$ and $\mathbf{y}$ are n- and r-dimensional vectors, respectively, $\mathbf{A}$, $\mathbf{B}$, $\mathbf{D}$, $\mathbf{E}$, $\mathbf{F}$ and $\mathbf{G}$ are matrices of technological coefficients, $\mathbf{c}$ and $\mathbf{d}$ are n and r-dimensional vectors of the objective function coefficients and $\mathbf{b}_1$, $\mathbf{b}_2$, $\mathbf{b}_3$ are ℓ_1-, ℓ_2-, and ℓ_3-dimensional right-hand side vectors. All vectors and matrices of coefficients are of appropriate dimension. An equivalent formulation of the above problem is

$$\begin{aligned} \text{P': Max } -z &= -\mathbf{cx} + \mathbf{dy} \\ \text{s.t.} \quad \mathbf{Ax} + \mathbf{By} &\le \mathbf{b}^1 \\ \mathbf{Dx} + \mathbf{Ey} &= \mathbf{b}^2 \\ -\mathbf{Fx} - \mathbf{Gy} &\le -\mathbf{b}^3 \\ \mathbf{x} \qquad &\ge \mathbf{0} \\ \mathbf{y} &\in \mathbb{R}^r \end{aligned}$$

Assigning vectors of dual variables $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$ to the respective sets of primal constraints, the associated dual problem is:

$$\begin{aligned} \text{P}_\text{D}\text{: Min } -z_D &= \mathbf{ub}_1 + \mathbf{vb}_2 - \mathbf{wb}_3 && \text{(Rule 5)} \\ \text{s.t.} \quad \mathbf{uA} + \mathbf{vD} - \mathbf{wF} &\ge -\mathbf{c} && \text{(Rule 3)} \\ \mathbf{uB} + \mathbf{vE} - \mathbf{wG} &= \mathbf{d} && \text{(Rule 4)} \\ \mathbf{u}, \qquad \mathbf{w} \quad &\ge \mathbf{0} && \text{(Rule 1)} \\ \mathbf{v} \qquad &\in \mathbb{R}^{\ell_2} && \text{(Rule 2)} \end{aligned}$$

The primal-dual relationships that were derived in Theorem 4.8 are now displayed in the synopsis in Table 4.2:

Table 4.2

P	P_D
finite optimal solution	finite optimal solution
unbounded "optimal" solutions	no feasible solution
no feasible solution	either unbounded "optimal" solutions or no feasible solutions

Furthermore, note that the dual of the dual problem is the primal problem, i.e., $(P_D)_D = P$. Examples for the primal-dual relations in the above synopsis are shown below.

Example 1: Both problems have finite optimal solutions.

P: Max $z = 2x_1 + x_2$
s.t. $x_1 \le 1$
$x_2 \le 1$
$x_1, \; x_2 \ge 0$

P_D: Min $z_D = u_1 + u_2$
s.t. $u_1 \ge 2$
$u_2 \ge 1$
$u_1, \; u_2 \ge 0$

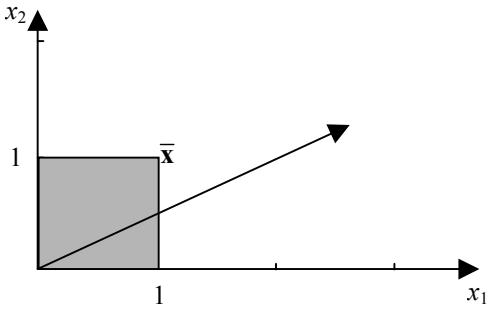


Figure 4.2a

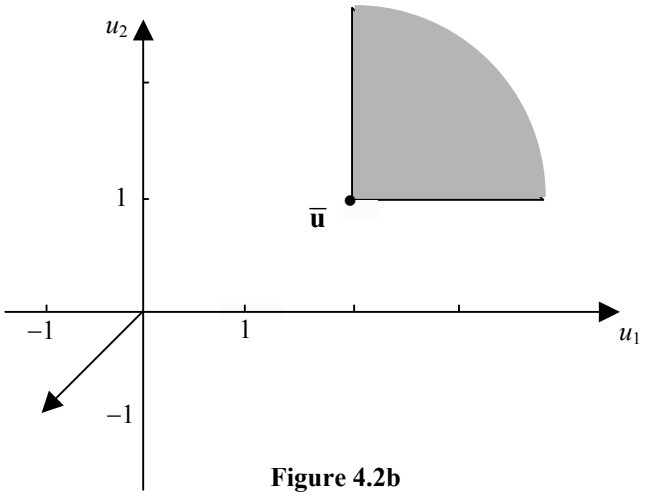


Figure 4.2b

The graphical representations of the problems P and P_D are shown in Figures 4.2a and 4.2b, respectively. The optimal solutions are indicated as $\bar{\mathbf{x}}$ and $\bar{\mathbf{u}}$, respectively. The optimal solutions are $\bar{\mathbf{x}} = [1, 1]^T$ with $\bar{z} = \mathbf{c}\bar{\mathbf{x}} = 3$ and $\bar{\mathbf{u}} = [2, 1]$ with $\bar{z}_D = \bar{\mathbf{u}}\mathbf{b} = 3$, respectively.

Example 2: P has no feasible solution and P_D has unbounded "optimal" solutions.

P: Max $z = x_1 + x_2$
s.t. $-x_1 + x_2 \geq 1$
$x_1 - x_2 \geq 1$
$x_1, \ x_2 \geq 0$

P_D: Min $z_D = u_1 + u_2$
s.t. $-u_1 + u_2 \leq 1$
$u_1 - u_2 \leq 1$
$u_1, \ u_2 \geq 0.$

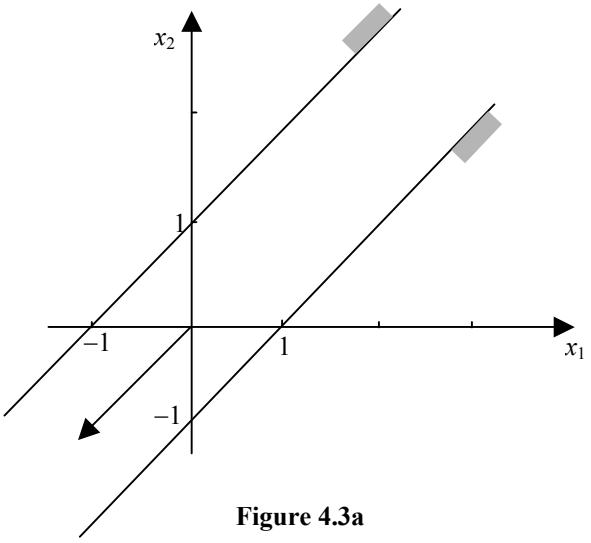


Figure 4.3a

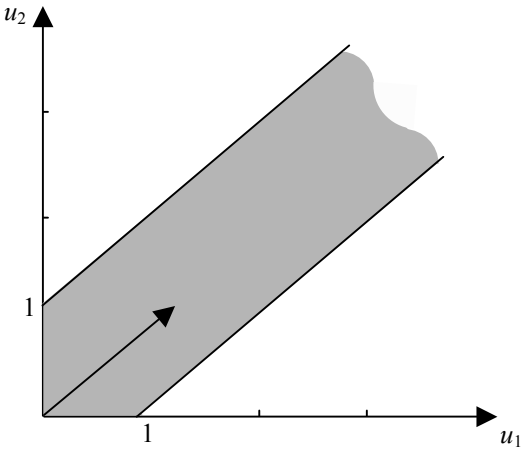


Figure 4.3b

The graphical representations of the problems in Figures 4.3a and 4.3b indicate that the primal problem has not feasible solution, while the dual problem has unbounded "optimal" solutions.

Example 3: Both problems have no feasible solutions.

P: Max $z = x_1 + x_2$
s.t. $x_1 - x_2 \le 1$
$x_1 - x_2 \ge 2$
$x_1,\ x_2 \ge 0$

P_D: Min $z_D = u_1 - 2u_2$
s.t. $u_1 - u_2 \ge 1$
$-u_1 + u_2 \ge 1$
$u_1,\ u_2 \ge 0$

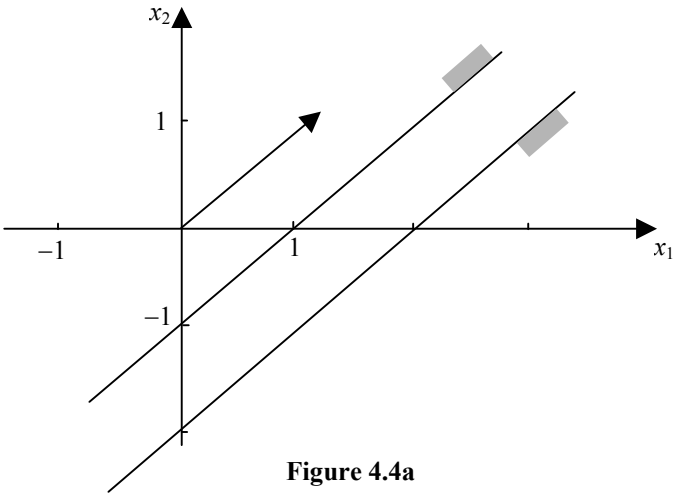


Figure 4.4a

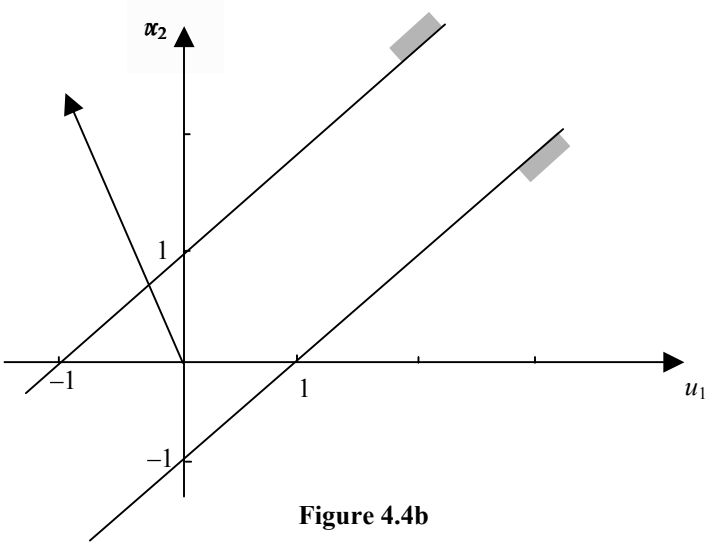


Figure 4.4b

Figures 4.4a and 4.4b represent the respective primal and dual problem formulation above. It is apparent that neither has a feasible solution.

4.2 Primal-Dual Relations

Although primal-dual relationships were theoretically developed in the first section of this chapter, the following paragraphs will take a close look at these relations from a computational point of view. In particular, we will show that if any primal problem has been solved with the simplex method, the dual solution, i.e., the values of the dual variables, can be found in the primal tableau. At this point, we will restrict ourselves to optimal primal and dual solutions, this restriction will be dropped at the end of this section.

Recall that at least one variable was added to every primal constraint, *viz*., a slack, excess and/or artificial variable, depending upon the type of the constraint. Now a pair of variables can be assigned to every primal constraint; the dual variable assigned to this constraint and the slack, excess or artificial variable added to the left-hand side of the primal constraint. The assignment of these pairs to the various constraint is summarized in Table 4.3:

Table 4.3

Primal constraint/variable	Variable pair
$\mathbf{a}_{i\bullet}\mathbf{x} \leq b_i$	(u_i, S_i)
$\mathbf{a}_{i\bullet}\mathbf{x} \geq b_i$	(u_i, E_i)
$\mathbf{a}_{i\bullet}\mathbf{x} = b_i$	(u_i, A_i)
$\mathbf{x}_j \geq 0$	(S_j^D, x_j) or (E_j^D, x_j)

These pairs indicate exactly where the value of a dual variable will be found in the optimal primal tableau: In case of a ≤ inequality, the optimal value of the dual variable $\overline{u}_i$ is found in the objective function row under the primal slack variable S_i; in case of a ≥ inequality, the optimal value of the dual variable $\overline{u}_i$ is found under the primal excess variable E_i, and in case of an equality, the value of $\overline{u}_j$ (which is clearly unrestricted since it was assigned to a primal equality) is found under the primal artificial variable A_i in the objective function row. Moreover, the dual slack and excess variables S_j^D and E_j^D will be found in the primal tableau under the primal decision variable x_j. For the sake of simplicity, assume that the given primal problem is in canonical form, i.e.,

$$\begin{array}{lll} \text{P: Max } z = \mathbf{cx} & & \text{P}_\text{D}\text{: Min } z_D = \mathbf{ub} \\ \quad \text{s.t.} \quad \mathbf{Ax}\ R\ \mathbf{b} & & \quad \text{s.t.} \quad \mathbf{uA}\ R\ \mathbf{c} \\ \quad \mathbf{x}\ R\ \mathbf{0} & & \quad \mathbf{u}\ R\ \mathbf{0} \end{array}$$

Here, R denotes any relation of the type ≤, ≥, or =. The optimal values of the primal variables x, S, E, and A are then found on the right-hand side value of the primal tableau (in case they are basic; nonbasic variables are zero by definition) as well as in the objective function row of the optimal dual tableau. The optimal

values of the primal decision variables are found under the dual slack, excess, and artificial variables, while the optimal values of the primal slack, excess, and artificial variables are found under the dual decision variables. Similarly, the optimal values of the dual decision variables are found in the objective row of the optimal primal tableau under the primal slack, excess, and artificial variables, while the optimal values of the dual slack, excess, and artificial variables are found in the objective row of the primal tableau under the primal decision variables. This situation is summarized in Table 4.4 for the primal and dual tableaus, respectively.

Table 4.4

Optimal primal tableau:

$\mathbf{x}$	$\mathbf{S}, \mathbf{E}, \mathbf{A}$	1
		$\overline{\mathbf{x}}\ \overline{\mathbf{S}}, \overline{\mathbf{E}}, \overline{\mathbf{A}}$
$\overline{\mathbf{S}}^D, \overline{\mathbf{E}}^D, \overline{\mathbf{A}}^D$	$\overline{\mathbf{u}}$	$\bar{z}$

Optimal dual tableau:

$\mathbf{u}$	$\mathbf{S}^D, \mathbf{E}^D, \mathbf{A}^D$	1
		$\overline{\mathbf{u}}, \overline{\mathbf{S}}^D, \overline{\mathbf{E}}^D, \overline{\mathbf{A}}^D$
$\overline{\mathbf{S}}, \overline{\mathbf{E}}, \overline{\mathbf{A}}$	$\overline{\mathbf{x}}$	$\bar{z}_D$

Example: Consider the following (primal) linear programming problem.

$$
\begin{aligned}
&\text{P: Max } z = 2x_1 + x_2 \\
&\text{s.t. } 5x_1 + 2x_2 \le 10 && (u_1, S_1) \\
&\quad 4x_1 + \ x_2 \ge \ 4 && (u_2, E_2) \\
&\quad -x_1 + 2x_2 = \ 1 && (u_3, A_3) \\
&\quad x_1, \quad x_2 \ge 0,
\end{aligned}
$$

where the pair of variables on the right indicate the dual variable that is associated with the constraint and the primal slack/excess/artificial variable, under which its value is found in the primal tableau. (In case the artificial variables were dropped from the tableau, the value of $-u_3$ would be found in the objective row of the primal problem under E_3). Similarly, the optimal values of x_1 and x_2 will be found in the optimal dual tableau under the dual excess variables E_1^D and E_2^D.

The optimal tableau for the primal problem above is

$T_{\text{primal}}^{\text{opt}}$:

x_1	x_2	S_1	E_2	A_2	A_3	1
0	0	$\frac{3}{4}$	1	–1	$-\frac{1}{4}$	3¼
1	0	$\frac{1}{6}$	0	0	$-\frac{1}{6}$	1½
0	1	$\frac{1}{12}$	0	0	$\frac{5}{12}$	1¼
0	0	$\frac{5}{12}$	0	0	$\frac{1}{12}$	4¼

The optimal primal solution is $\bar{\mathbf{x}} = (1\frac{1}{2}, 1\frac{1}{4})^T$ with $\bar{z} = 4\frac{1}{4}$. (1½, 1¼)^T According to the above pairs of variables, the dual solution is $\bar{\mathbf{u}} = (\frac{5}{12}, 0, \frac{1}{12})$ with $\bar{z}_D = 4\frac{1}{4}$. Furthermore, we note that $\bar{E}_1^D = \bar{E}_2^D = 0$, indicating that both dual constraints are satisfied as equations at optimum.

The dual problem of the above linear programming problem P can be formulated as

$$
\begin{aligned}
\mathrm{P_D}\text{: Min } z_D = {} & 10u_1 - 4u_2 + u_3 & \\
\text{s.t.} \quad & 5u_1 - 4u_2 - u_3 \geq 2 & (E_1^D, x_1) \\
& 2u_1 - u_2 + 2u_3 \geq 1 & (E_2^D, x_2) \\
& u_1, \quad u_2 \quad \geq 0 & \\
& u_3 \in \mathbb{R}, &
\end{aligned}
$$

where the pairs on the right indicate where to find the values of the primal variables x_1 and x_2 in the dual tableau (*viz.*, underneath the variables E_1^D and E_2^D). After replacing the unrestricted variable u_3 by the difference of two nonnegative variables u_3^+ and u_3^- and solving the problem, we obtain the optimal dual tableau

$T_{\text{dual}}^{\text{opt}}$:

u_1	u_2	u_3^+	u_3^-	E_1^D	E_2^D	A_1^D	A_2^D	1
1	$-\frac{3}{4}$	0	0	$-\frac{1}{6}$	$-\frac{1}{12}$	$\frac{1}{6}$	$\frac{1}{12}$	$\frac{5}{12}$
0	$\frac{1}{4}$	1	–1	$\frac{1}{6}$	$-\frac{5}{12}$	$-\frac{1}{6}$	$\frac{5}{12}$	$\frac{1}{12}$
0	3¼	0	0	1½	1¼	–1½	–1¼	–4¼

which contains exactly the same information as the optimal primal tableau. The above pairs of variables can be used again to find the optimal primal solution; e.g., the optimal value of S_1 is found under u_1, the optimal value of E_2 is found under u_2 and so forth, and finally the optimal values of x_1 and x_2 are found under E_1^D and E_2^D, respectively. Loosely speaking, the information on the right-hand side of the primal tableau is found in the objective function row in the dual tableau and vice versa. This also explains the term dual degeneracy, which occurs if a zero appears on the right-hand side of a dual tableau, corresponding to a zero under a primal nonbasic variable in a primal tableau.

Clearly, if artificial variables are dropped from the tableau whenever they become nonbasic, the information concerning dual variables gets lost. The values of the dual variables may be reconstructed, though, based on duality relations.

Suppose now that, as is usually the case, the artificial variables are dropped form the tableau as soon as they leave the basis; i.e. if at least one primal feasible solution exists and the optimal primal tableau is nondegenerate, then no artificial variables are left in the tableau. In those cases, the optimal values of the dual variables, assigned to primal equalities, can not be read directly form the optimal primal tableau, but it is possible to reconstruct these values by using the values of the dual slack and excess variables. As an example, assume that the optimal primal tableau for the above problem dose not include the A_2 and A_3 columns. Hence it is only known that $\bar{u}_1 = \frac{5}{12}$ and $\bar{u}_2 = 0$. Since, however, $\bar{E}_1^D = 0$, we know that the first dual constraint if fulfilled as equality, i.e., $5u_1 - 4u_2 - u_3 = 2$; using the optimal values of u_1 and u_2 yields $\bar{u}_3 = \frac{1}{12}$ which is the desired result. (Note that the equality $\mathbf{c}\bar{\mathbf{x}} = \bar{\mathbf{u}}\mathbf{b}$ can also be used to reconstruct the value of one dual variable).

Finally, one more remark in this context: Since optimal primal and dual tableaus contain the same information, either problem could be solved. For manual computations, it is usually easier to solve the primal problem if $n > m$, while the dual problem has a smaller tableau, if $n < m$. These considerations are, however, irrelevant for automatic computations.

In some instances, it is more advantageous to solve the dual problem rather than the primal. Consider a primal problem in canonical form with n variables and m constraints. After adding slack variables, each primal tableau includes a total of $m + n$ variables (columns) and m constraints (rows), not counting right-hand sides and objective function, for a total of $m(m + n)$ elements. The dual, however, has m problem variables and n constraints (in the worst case all dual variables are unrestricted and each of those variables has to be replaced by two nonnegative variables and all constraints are of type ≥) so the dual has no more than $2(m + n)$ variables and n constraints.

In the following we will have another look at the primal simplex method in view of the optimality criteria developed in this chapter, i.e., primal feasibility, dual feasibility, and (weak) complementary slackness conditions. Primal feasibility is achieved in a primal simplex tableau, if no artificial variable with a value greater than zero is left. On the other hand, dual feasibility is obtained, if all entries in the given objective function of the primal tableau are nonnegative. If, say, $c_j < 0$, for some primal variable x_j, then currently the dual variable $E_j^D < 0$; if $c_i < 0$ in a primal simplex tableau under some variable S_i, then $u_i < 0$; both cases violate feasibility of the dual solution. For the complementary slackness conditions, consider the following pair of primal and dual problems:

$$\begin{array}{lll} \text{P: Max } z = \mathbf{cx} & & \text{P}_\text{D}\text{: Min } z_D = \mathbf{ub} \\ \quad \text{s.t.} \quad \mathbf{Ax} \le \mathbf{b} & & \quad \text{s.t.} \quad \mathbf{uA} \ge \mathbf{c} \\ \quad \mathbf{x} \ge \mathbf{0} & & \quad \mathbf{u} \ge \mathbf{0} \end{array}$$

Adding a slack variable to the left-hand side of the i-th primal constraint results in $\mathbf{a}_{i\bullet}\mathbf{x} + S_i = b_i$ or, equivalently, $S_i = b_i - \mathbf{a}_{i\bullet}\mathbf{x}$; subtraction of an excess variable from the left-hand side of the j-th dual constraint yields $\mathbf{ua}_{\bullet j} - E_j^D = c_j$ or $E_j^D = \mathbf{ua}_{\bullet j} - c_j$. Then the two complementary slackness conditions $u_i(\mathbf{a}_{i\bullet}\mathbf{x} - b_i) = 0$ and $(\mathbf{ua}_{\bullet j} - c_j)\, x_j = 0$ can be rewritten as $u_i S_i = 0$ and $E_j^D x_j = 0$, respectively.

These two conditions are satisfied if at least one of the two variables in each condition equals zero. Consider $u_i S_i = 0$. Clearly, if $S_i = 0$, this condition is satisfied. Suppose now that $S_i > 0$. This is only possible if S_i a basic variable which—by definition—has a zero coefficient in the objective function; that is exactly the current value of u_i and so $u_i = 0$. An equivalent argument can be applied to the second complementary slackness condition. In other words: during the iterations with the primal simplex method, the complementary slackness

$$O \xrightarrow[\text{phase 1}]{} \text{primal feasible solution} \xrightarrow[\text{phase 2}]{} \text{optimal solution}$$

Figure 4.5

condition is always fulfilled. Now the simplex method can be seen as a algorithm which generates points (solutions), the first one being the origin *0* in the $x_1, x_2, \ldots, x_n$ space of the original nonbasic variables; after some iterations the first feasible solution is reached and after some more iterations the optimal solution is obtained. This can be visualized in Figure 4.5.

Table 4.5

	Primal feasibility	Dual feasibility	Complementary slackness
Phase 1	No	(usually) No	Yes
Phase 2	Yes	No	Yes
Optimal solution	Yes	Yes	Yes

Table 4.5 summarizes which of the three conditions are fulfilled in Phase 1 (including initial solutions *0* but excluding the first feasible solution), Phase 2 (including the first feasible solution but not the optimal solution), and at the optimal solution.

We have seen that every simplex tableau displays one primal and its corresponding dual solution. Moreover, in the absence of primal and dual degeneracy, a unique optimal solution exists for both problems with primal and dual objective function values being equal, all other primal feasible solutions will have a smaller z-value whereas all dual feasible solutions will have a larger z_D-value. This is displayed in Figure 4.6.

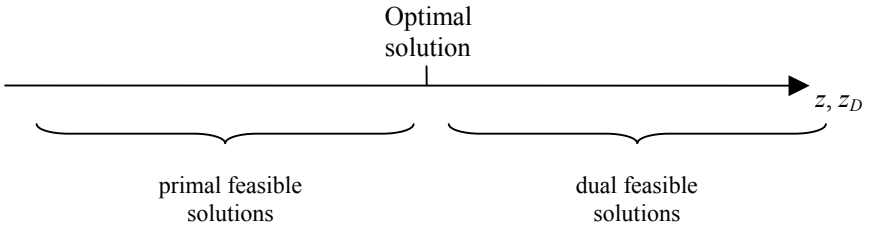


Figure 4.6

Example: Consider the following pair of primal and dual linear programming problems:

P: Max $z = x_1 + x_2$
s.t. $2x_1 + x_2 \le 4$
$2x_1 + 3x_2 \le 6$
$x_1, x_2 \ge 0$

P_D: Min $z_D = 4u_1 + 6u_2$
s.t. $2u_1 + 2u_2 \ge 1$
$u_1 + 3u_2 \ge 1$
$u_1,\ u_2 \ge 0$

Table 4.6

Primal solution **(x, S)**	Primal basis	Primal feasible?	Dual solution **(u, $\mathbf{E}^D$)**	Dual basis	Dual feasible?	$z = z_D$
(0, 0, 4, 6)	S_1, S_2	Yes	(0, 0, −1, −1)	E_1^D, E_2^D	No	0
(0, 2, 2, 0)	x_2, S_1	Yes	(0, ⅓, −⅓, 0)	u_2, E_1^D	No	2
(2, 0, 0, 2)	x_1, S_2	Yes	(½, 0, 0, −½)	u_1, E_2^D	No	2
(1½, 1, 0, 0)	x_1, x_2	Yes	$(\frac{1}{4}, \frac{1}{4}, 0, 0)$	u_1, u_2	Yes	2½
(3, 0, −2, 0)	x_1, S_1	No	(0, ½, 0, ½)	u_2, E_2^D	Yes	3
(0, 4, 0, −6)	x_2, S_2	No	(1, 0, 1, 0)	$u_1,\ E_1^D$	Yes	4

Table 4.6 displays all primal basic solutions and their dual counterparts, along with information regarding primal and dual feasibility as well as the values of the objective function.

The primal-dual relationships can also be given an interesting graphical visualization. Suppose that a primal linear programming problem with n decision variables and m constraints is given; including the necessary m slack variables leads to a total of $n + m$ variables and m equalities. Every basis then includes m basic variables and hence n nonbasic variables; therefore a total of at most

$\binom{m+n}{m}$ different bases (if not all $[m \times m]$-dimensional submatrices have full rank, there will be less). Suppose all $\binom{n+m}{m}$ bases exist; then we could select any m variables, designate them as basic variables and obtain the solution for this basis by setting the remaining n (nonbasic) variables equal to zero and solving the system of simultaneous linear equations with the m basic variables only. This may result in a feasible solution or not. At the same time one could represent all basic variables in terms of the nonbasic variables and then the solution is always the origin in the nonbasic space. Depending on the status of the origin, we can distinguish between the four cases shown in Table 4.7.

Table 4.7

The origin is	The current solution is	
	Primal feasible?	Dual feasible?
(1) Feasible but not optimal	Yes	No
(2) Not feasible, but has better z-value than any feasible point	No	Yes
(3) Not feasible, feasible points with better z-value exist	No	No
(4) Feasible and optimal	Yes	Yes

Note that in case 4, the gradient in primal and dual nonbasic spaces are directed into the negative orthant. A similar synopsis can be set up for the dual problem. This can be illustrated by the following

Example: Consider the following pair of dual linear programs, in which the necessary slack and excess variables have already been added:

$$\begin{aligned} \text{P: Max } z &= 2x_1 + x_2 \\ \text{s.t. } & 2x_1 + 3x_2 + S_1 = 12 \\ & 4x_1 + x_2 + S_2 = 8 \\ & x_1, \quad x_2, \quad S_1, S_2 \geq 0 \end{aligned}$$

and

$$\begin{aligned} \text{P}_\text{D}\text{: Min } z_D &= 12u_1 + 8u_2 \\ \text{s.t. } & 2u_1 + 4u_2 - E_1^D = 2 \\ & 3u_1 + u_2 - E_2^D = 1 \\ & u_1, \quad u_2, \quad E_1^D, E_2^D \geq 0. \end{aligned}$$

Each of the above problems has a total of four variables and two equations and the $\binom{2+2}{2} = 6$ bases do exist. In the graphs below, every possible primal and its corresponding dual basis is enumerated; primal and dual problems are rewritten in terms of nonbasic variables and graphed, and the corresponding primal tableau is displayed.

Primal basis: (S_1, S_2)
Primal nonbasis: (x_1, x_2)

P: Max $z = 2x_1 + x_2$
s.t. $2x_1 + 3x_2 \le 12$
$4x_1 + x_2 \le 8$
$x_1, \quad x_2 \ge 0$

Dual basis : (E_1^D, E_2^D)
Dual nonbasis: (u_1, u_2)

P_D: Min $z_D = 12u_1 + 8u_2$
s.t. $2u_1 + 4u_2 \ge 2$
$3u_1 + u_2 \ge 1$
$u_1, \quad u_2 \ge 0$

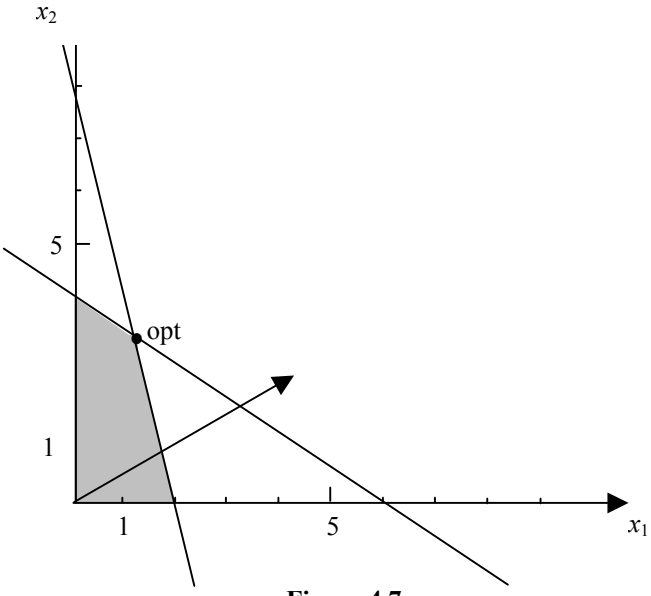


Figure 4.7a

Figures 4.7a and 4.7b show the primal and the dual problem at this point.

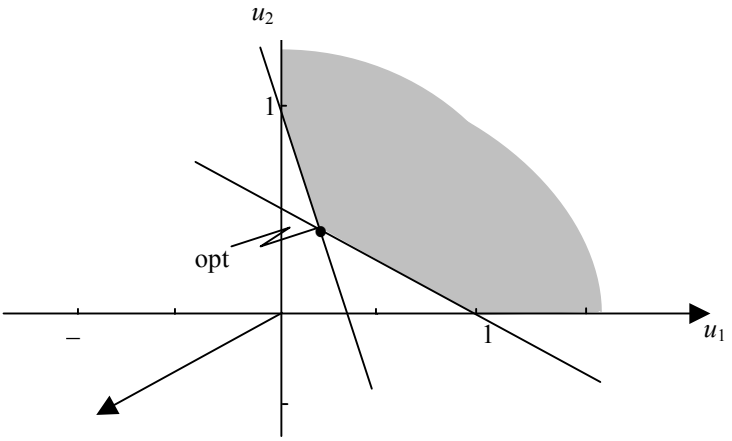


Figure 4.7b

The graphs in Figures 4.7a and 4.7b indicate that the current solution is primal feasible but not dual feasible. This is case (1). Another basis is:

Primal basis: (x_2, S_2)	Dual basis : $(u_1,\ E_1^D)$
Primal nonbasis: (x_1, S_1)	Dual nonbasis: $(u_2,\ E_2^D)$

P: Max $z = \frac{4}{3}x_1 - \frac{1}{3}S_1 + 4$	P_D: Min $z_D = 4u_2 + 4E_2^D + 4$
s.t. $\frac{2}{3}x_1 + \frac{1}{3}S_1 \le 4$	s.t. $-\frac{1}{3}u_2 + \frac{1}{3}E_2^D \ge -\frac{1}{3}$
$\frac{10}{3}x_1 - \frac{1}{3}S_1 \le 4$	$\frac{10}{3}u_2 + \frac{2}{3}E_2^D \ge \frac{4}{3}$
$x_1, \quad S_1 \ge 0$	$u_2, \quad E_2^D \ge 0$

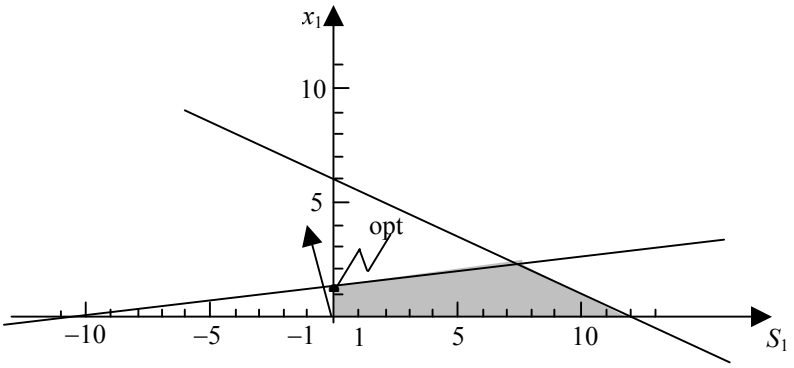


Figure 4.7c

The graph in Figure 4.7c displays the primal problem at this point, while Figure 4.7d shows the dual problem.

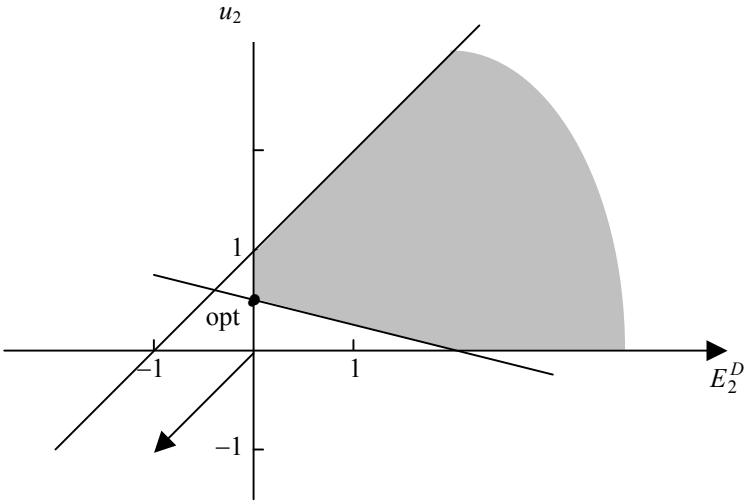


Figure 4.7d

Clearly, the origin in the "primal graph" in Figure 4.7c is (primal) feasible but not optimal (case 1). The dual graph in Figure 4.7d indicates that the origin is not (dual) feasible but has a lower z-value than any feasible dual point, hence the primal solution must be feasible.

The next basis to be examined is as follows.

Primal basis: (x_2, S_1)	Dual basis : (u_2, E_1^D)
Primal nonbasis: (x_1, S_2)	Dual nonbasis: (u_1, E_2^D)

P: Max $z = -2x_1 - S_2 + 8$	P_D: Min $z_D = -12u_1 + 8E_2^D + 8$
s.t. $-10x_1 - 3S_2 \le -12$	s.t. $-3u_1 + E_2^D \ge -1$
$4x_1 + S_2 \le 8$	$-10u_1 + 4E_2^D \ge -2$
$x_1, \quad S_2 \ge 0$	$u_1, \quad E_2^D \ge 0$

Figure 4.7e is the primal problem at this basis, while Figure 4.7f shows the dual problem.

The primal graph in Figure 4.7e indicates that the current solution is not primal feasible but since the origin has a higher z-value than any primal feasible point, it is dual feasible. The present dual solution shown in Figure 4.7f indicates that while the present solution is feasible, it is not yet optimal.

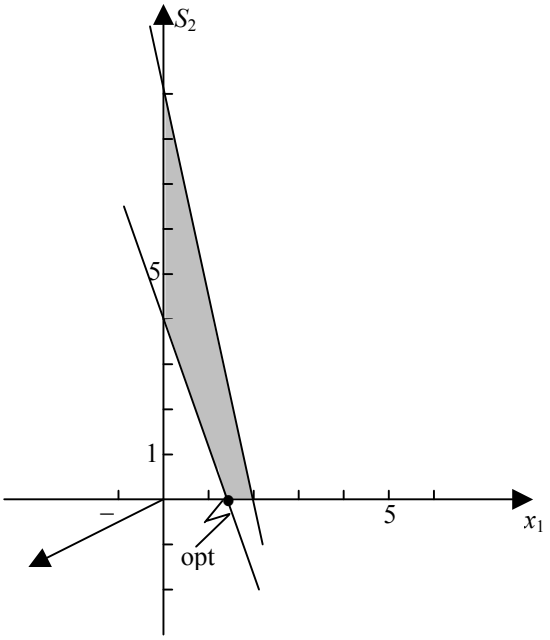


Figure 4.7e

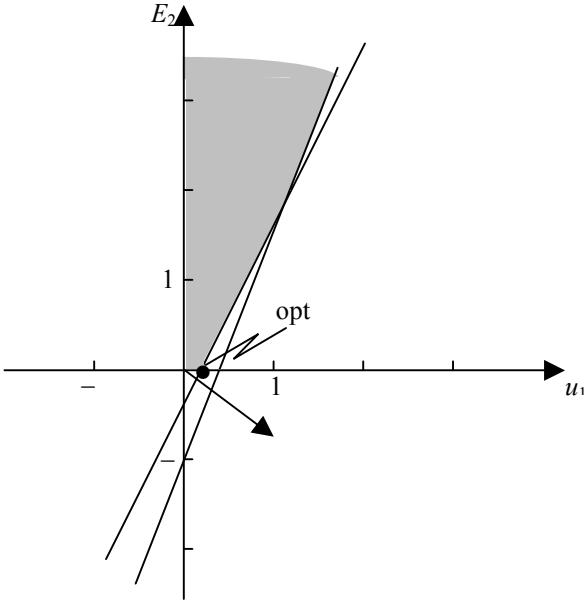


Figure 4.7f

The next pair of dual bases is given as

Primal basis: (x_1, S_2)	Dual basis : (u_1, E_2^D)
Primal nonbasis: (x_2, S_1)	Dual nonbasis: (u_2, E_1^D)

P: Max $z = -2x_2 - S_1 + 12$

s.t $\frac{3}{2}x_2 + \frac{1}{2}S_1 \le 6$

$-5x_2 - 2S_1 \le -16$

$x_2, \quad S_1 \ge 0$

P_D: Min $z_D = -16u_2 + 6E_1^D + 12$

s.t. $-5u_2 + \frac{3}{2}E_1^D \ge -2$

$-2u_2 + \frac{1}{2}E_1^D \ge -1$

$u_2, \quad E_1^D \ge 0$

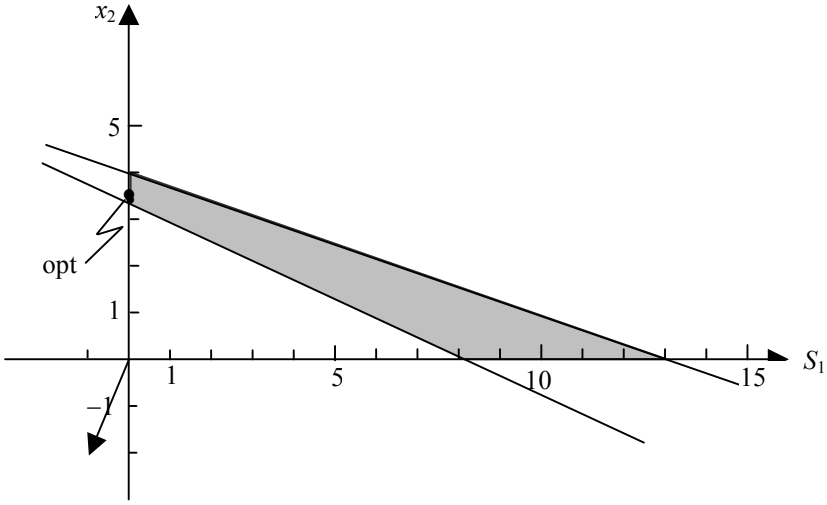


Figure 4.7g

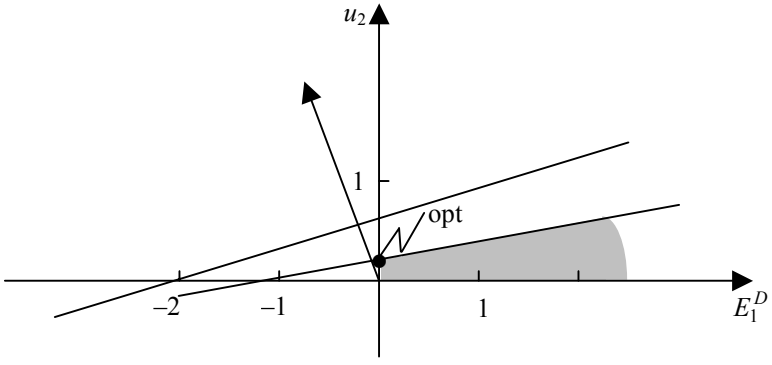


Figure 4.7h

Again, Figure 4.7g represents the primal problem at the present basis, while Figure 4.7h is the dual problem at this point. Figure 4.7g indicates primal infeasibility of the current solution but also shows that the origin has a higher z-value than any primal feasible point (case 2). Hence the corresponding dual solution in Figure 4.7h is feasible.

The next basis is as follows.

Primal basis: (x_1, S_1) Dual basis : $(u_2,\ E_2^D)$

Primal nonbasis: (x_2, S_2) Dual nonbasis: $(u_1,\ E_1^D)$

P: Max $z = \frac{1}{2}x_2 - \frac{1}{2}S_2 + 4$

s.t. $\frac{5}{2}x_2 - \frac{1}{2}S_2 \le 8$

$\frac{1}{4}x_2 + \frac{1}{4}S_2 \le 2$

$x_2, \quad S_2 \ge 0$

P_D: Min $z_D = 8u_1 + 2E_1^D + 4$

s.t. $\frac{5}{2}u_1 + \frac{1}{4}E_1^D \ge \frac{1}{2}$

$-\frac{1}{2}u_1 + \frac{1}{4}E_1^D \ge -\frac{1}{2}$

$u_1, \quad E_1^D \ge 0$

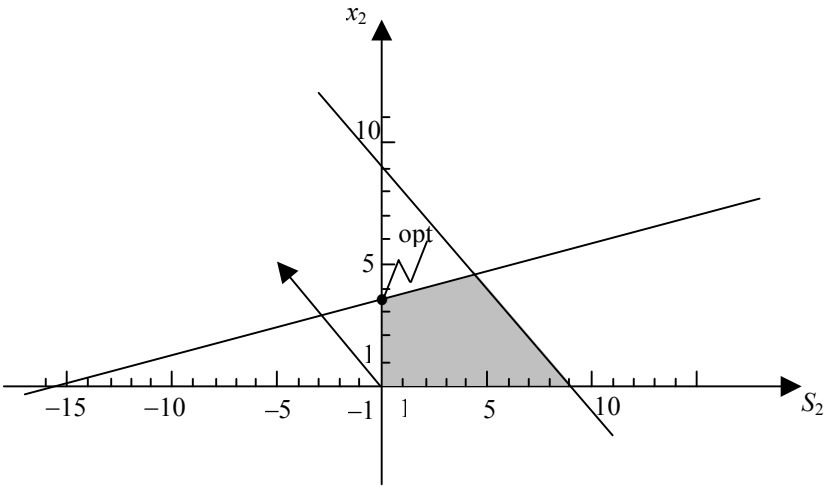


Figure 4.7i

Figure 4.7i shows the primal problem at the present basis. Figure 4.7j represents the dual solution at the present basis.

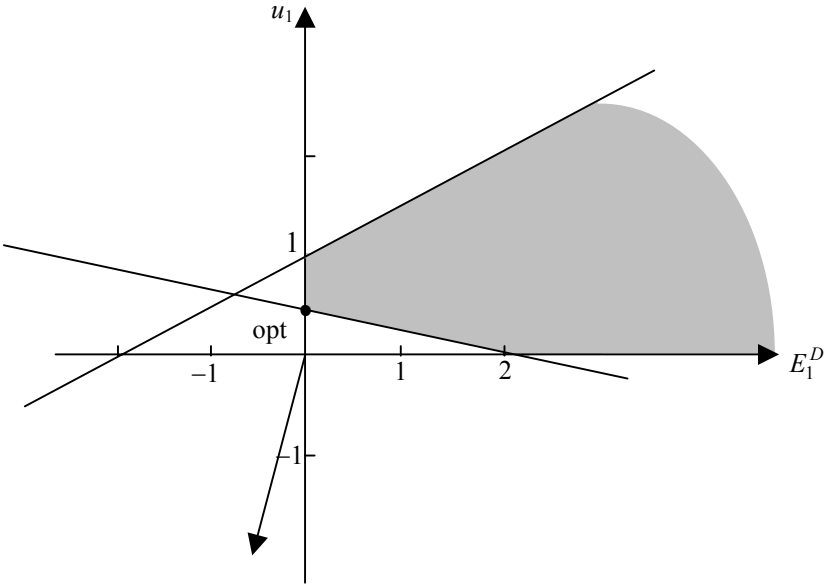


Figure 4.7j

The graphs in Figures 4.7i and 4.7j exhibit primal feasibility as well as dual infeasibility.

Finally, the last possible basis is:

Primal basis: (x_1, x_2)	Dual basis : (u_1, u_2)
Primal nonbasis: (S_1, S_2)	Dual nonbasis: (E_1^D, E_2^D)

P: Max $z = -\frac{1}{5}S_1 - \frac{2}{5}S_2 + 5\frac{3}{5}$	P_D: Min $z_D = \frac{6}{5}E_1^D + \frac{16}{5}E_2^D + 5\frac{3}{5}$
s.t. $\frac{2}{5}S_1 - \frac{1}{5}S_2 \le \frac{16}{5}$	s.t. $-\frac{1}{10}E_1^D + \frac{2}{5}E_2^D \ge -\frac{1}{5}$
$-\frac{1}{10}S_1 + \frac{3}{10}S_2 \le \frac{6}{5}$	$\frac{3}{10}E_1^D - \frac{1}{5}E_2^D \ge -\frac{2}{5}$
$S_1, S_2 \ge 0$	$E_1^D, \quad E_2^D \ge 0$

Figure 4.7k shows the primal problem at the present basis, while Figure 4.7l displays the dual problem.

In the situation shown in Figures 4.7k and 4.7l, both gradients of the objective functions are directed into the negative quadrant. At both origins primal feasibility, dual feasibility and complementary slackness is fulfilled, so that an optimal solution has been found, which is Case 4 in Table 4.7.

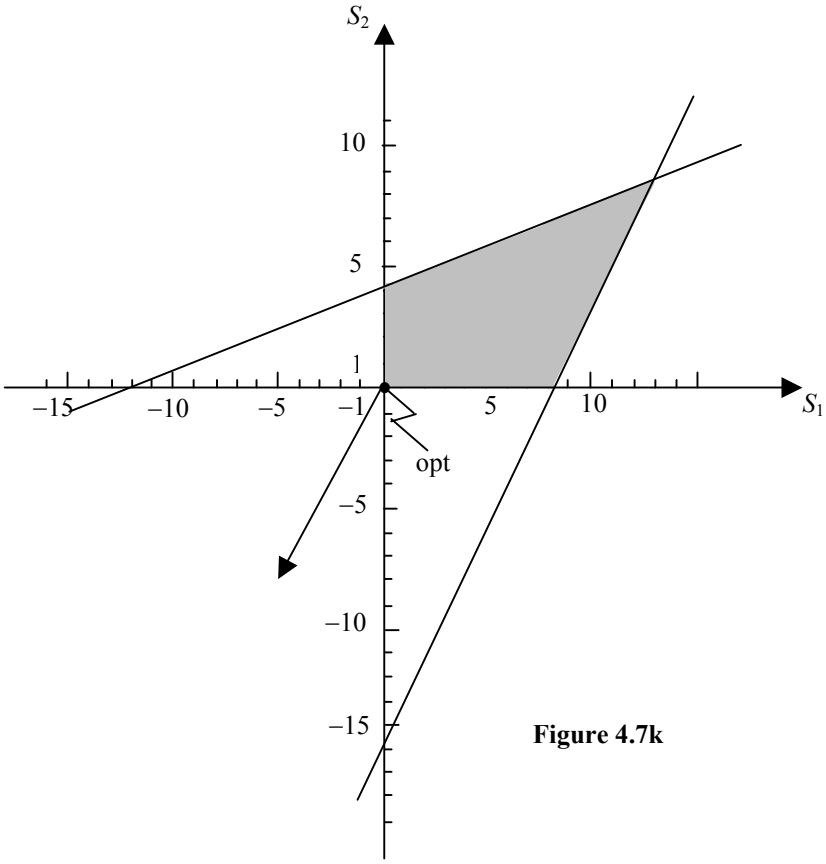


Figure 4.7k

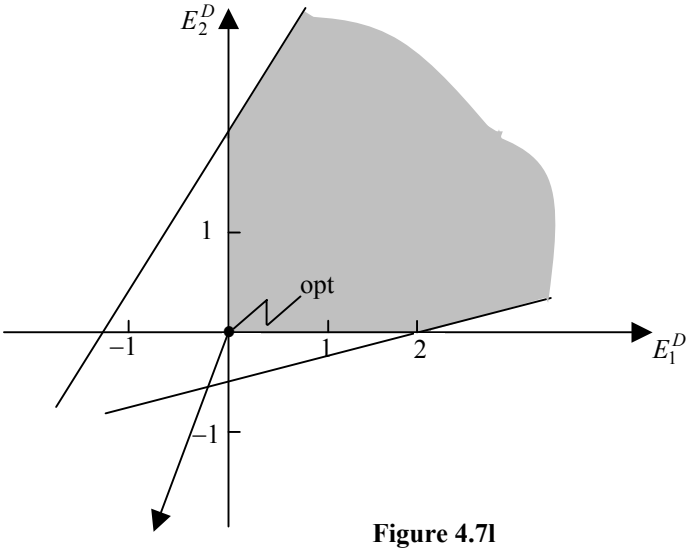


Figure 4.7l

4.3 Interpretations of the Dual Problem

This section will describe interpretations of the dual variables in some standard problems that have been formulated in Chapter 2 of this volume. For this purpose, we will consider three examples. The first is a single-variable production problem that may provide the general flavor of the interpretation of the dual problem, the second is a small diet problem, and the last model we formulate, dualize, and discuss in this section is a standard transportation problem.

Example 1: Consider a company that manufactures a single product by means of a single machine. The unit selling price of the product is \$5, all units that are made can also be sold, it takes 2 hours on the machine to manufacture one unit of the product, and the capacity of the machine is 8 hours. Denoting by x the number of units of the product that we will manufacture and sell, we can then formulate the revenue-maximizing problem as

$$\begin{aligned} \text{P: Max } & z = 5x \\ \text{s.t.} \quad & 2x \le 8 \\ & x \ge 0. \end{aligned}$$

The parameters and the variable are then measured in the following units: The price 5 is expressed in [\$/hour], the machine capacity 8 is measured in [hours], the time consumption for the manufacturing of the product is 2 [hour/quantity unit], and the variable x is measured in [quantity units].

The optimal solution of this primal problem is $\bar{x} = 4$ with a revenue of $\bar{z} = 20$. The dual of the above problem can be formulated as

$$\begin{aligned} \text{P}_\text{D}\text{: Min } & z_D = 8u \\ \text{s.t.} \quad & 2u \ge 5 \\ & u \ge 0, \end{aligned}$$

where the dual variable u is measured in [\$/hour]. The optimal dual solution is $\bar{u} = 2½$ with $\bar{z}_D = 20$. Knowledge of the optimal values for the dual variables provides an "automatic" break-even analysis. In fact, the optimal value of the dual variable associated with the i-th constraint (resource) denotes the maximal price we should be prepared to pay for one unit of this resource. In this problem, we should not pay more than \$2.50 for every machine hour. In fact, if we pay exactly that amount for every respective unit of resource as indicated by the optimal value of its dual variable, then no losses and no profits would result. Here, paying \$2.50 per machine hour results in the return of $\bar{z}$ = \$20 being completely absorbed by the total cost of $\bar{z}_D$ = \$20.

Because of this interpretation, the optimal values of the dual variables are frequently called *shadow prices* or *opportunity costs*, depending on the particular

application. One of the early contributors to use shadow prices (albeit not in the context of linear programming) was Schmalenbach (1947).

Example 2: An individual wants to plan his diet, which consists of only pork and beans. The three nutrients that are considered in this problem are protein, vitamin E, and carbohydrates. Details concerning the nutritional content of the foodstuffs, their prices, and the required nutritional content are shown in Table 4.8:

Table 4.8

	Pork	Beans	Nutrients needed (at least)
Protein	5	2	300
Vitamin E	9	1	60
Carbohydrates	7	2	800
Unit selling price	\$2.50	\$0.30	

By denoting with x_1 and x_2 the respective number of servings of pork and beans, respectively, a cost-minimizing diet problem can be formulated as a linear programming problem as follows.

$$\text{P: Min } z = 2.5x_1 + 0.3x_2$$

$$\begin{aligned} \text{s.t.} \quad & 5x_1 + 2x_2 \geq 300 && \text{(protein)} \\ & 9x_1 + 1x_2 \geq 510 && \text{(Vitamin E)} \\ & 7x_1 + 2x_2 \geq 800 && \text{(carbohydrates)} \\ & x_1, \quad x_2 \geq 0. \end{aligned}$$

The optimal tableau of this problem is

T^{opt}:

x_1	x_2	S_1	S_2	S_3	1
0	1	0	$-\frac{7}{11}$	$-\frac{9}{11}$	330
1	0	0	$\frac{-2}{11}$	$\frac{1}{11}$	20
0	0	1	$-\frac{4}{11}$	$-\frac{13}{11}$	460
0	0	0	$\frac{29}{110}$	$\frac{2}{110}$	−149

The optimal primal solution is then to buy (and eat) 20 servings of pork and 330 servings of beans. The total cost of this diet are \$149.

Consider now the units of the variables and parameters used in the primal model. The variables x_j are expressed in the number of servings in the diet, the parameters c_j express the cost per serving, the right-hand side values b_i denote the quantity of nutrients included in the diet, and the technological coefficients a_{ij} denote the

quantity of nutrient i per serving of food j. The primal problem P was then the usual cost-minimizing problem that ensures that the planner has sufficient nutrients in the diet.

P: Min $z = \mathbf{cx}$
s.t. $\mathbf{Ax} \geq \mathbf{b}$
$\mathbf{x} \geq \mathbf{0}$.

Consider now the dual problem. Formally, we have

P_D: Max $z_D = \mathbf{ub}$
s.t. $\mathbf{uA} \leq \mathbf{c}$
$\mathbf{u} \geq \mathbf{0}$.

The dual variables are now measured in cost per unit of nutrient. Suppose now that a firm contacts the planner with the following offer. The firm is in the business of manufacturing pills that include some essential nutrients. This pill manufacturer is now determining a pricing system $\mathbf{u}$ for his business, where u_i denotes the amount he will charge for one unit of nutrient i that is included in his pills.

The pill manufacturer now makes the following offer to the diet planner. Instead of eating the foods under consideration, the suggestion is to buy pills instead. In order to make the offer "palatable," the pill manufacturer ensures with the dual constraints that each combination of nutrients in a pill are never more expensive than the food that includes the same quantity of nutrients. Given that, the manufacturer will attempt to maximize his profit.

This "competitive" system also allows a glimpse at the strong connection between linear programming in general (and duality in particular) on the one hand, and game theory on the other. For further details, readers are referred to Eiselt and Sandblom (2004).

Example 3: Consider a standard transportation problem (see Chapter 2) that can be formulated as

$$\text{P: Min } z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$$

$$\text{s.t. } \sum_{j=1}^{n} x_{ij} = s_i \quad \forall\ i = 1,...,m$$

$$\sum_{i=1}^{m} x_{ij} = d_j \quad \forall\ j = 1,...,n$$

$$x_{ij} \geq 0\ \forall\ i=1, \ldots, m; j=1, \ldots, n.$$

Assigning dual variables u_i to the first m constraints and dual variables v_j to the last n constraints, the dual problem can be formulated as

$$\begin{aligned} \text{P}_\text{D}\text{: Max } z_D &= \sum_{i=1}^{m} s_i u_i + \sum_{j=1}^{n} d_j v_j \\ \text{s.t. } u_i + v_j &\le c_{ij} \;\forall\; i=1, \ldots, m; j=1, \ldots, n \\ u_i, \quad v_j &\in \mathbb{R} \;\forall\; i=1, \ldots, m; j=1, \ldots, n. \end{aligned}$$

As usual, the primal problem has the parameters s_i and d_j as well as the variables x_{ij} measured in quantity units, and the unit costs c_{ij} expressed in [$/quantity units], so that the decision maker wants to minimize his costs subject to the constraints that all units in the warehouses are shipped out, and all customer demands at the destinations are satisfied (we assume that the transportation problem is balanced).

The dual problem relates to a carrier who offers to our decision maker to perform the transportation task for our decision maker. The carrier will devise a pricing system by which he charges a per-unit price of u_i at the point of departure and a per-unit price of v_j at the destination. The dual constraints ensure that the per-unit costs on each route do not exceed the cost to the decision maker if he were to transport the units himself, and the carrier's objective is to maximize his profits.

It is now also possible to establish the formal conditions that govern the existence of reshipments and overshipments as defined in Section 2.8 For that purpose, let $\bar{x}_{ij}$ $i = 1, \ldots, m; j = 1, \ldots, n$ denote optimal solutions to a primal transportation problem, and let the associated optimal dual solution be $u_i, i = 1,...,m$ and $v_n, j = 1,...,n$. Finke (1977) has then proved

Theorem 4.13: Reshipments can improve the optimal solution if and only if there exists at least one pair (i, j), such that. $u_i + v_j < -c_{ij}$. Overshipments can improve an optimal solution if and only if there exists at least one pair (i, j), such that $u_i + v_j < 0$.

As a numerical illustration, consider again the pertinent examples of Section 2.8.

Example 4: The problem has two origins with supplies of 10 units each, while the two destinations have demands of 5 and 10 units, respectively. The transportation costs are displayed in the matrix $\mathbf{C} = \begin{bmatrix} 2 & 6 \\ 1 & 2 \end{bmatrix}$ and the optimal transportation plan without reshipments is $T = \begin{bmatrix} 5 & 5 \\ 0 & 10 \end{bmatrix}$, and the optimal dual solution is $\bar{\mathbf{u}}$ = [2, –2] and $\bar{\mathbf{v}}$ = [0, 4]. Here, $\bar{u}_2 + \bar{v}_1 = -2 < -1 = -c_{21}$, indicating the possibility of improving the solution by means of reshipments.

Example 5: The two origins have supplies of 10 units each, while there are demands of 15 and 5 at the two destinations. The transportation costs are shown in the matrix $\mathbf{C} = \begin{bmatrix} 1 & 4 \\ 4 & 1 \end{bmatrix}$, and the optimal transportation plan of the model without overshipments is $\overline{T} = \begin{bmatrix} 10 & 0 \\ 5 & 5 \end{bmatrix}$. The associated solution of the dual problem is $\overline{\mathbf{u}}$ = [1, 4] and $\overline{\mathbf{v}}$ = [0, −3]. It is apparent that $\overline{u}_1 + \overline{v}_2 = 1 - 3 = -2 < 0$, so that Theorem 4.13 indicates that overshipments will be able to improve the solution.

5 EXTENSIONS OF THE SIMPLEX METHOD

This chapter will introduce a number of extensions to the standard (primal) simplex method discussed in Chapter 3 of this volume. The first section introduces a technique that allows variables to be constraint by upper bounds without introducing these bounds in the tableau explicitly, thus keeping the tableaus small. The second section describes the column-generation technique, another method that has proven to be an invaluable tool for large-scale problem. Finally, the chapter concludes with the description of the dual simplex method, a technique that works on the primal problem but proceeds differently from the primal simplex method, in that it retains dual feasibility at all times, while trying to achieve primal feasibility. This technique is used extensively in integer programming as well as in methods that add constraints during the solution process.

5.1 The Dual Simplex Method

The roots of the dual simplex method date back to the work by Lemke (1954). Recall that the primal simplex method described in Chapter 3 of this volume maintains primal feasibility and complementary slackness throughout its execution of Phase 2, the method will terminate once dual feasibility is reached. In contrast, the dual simplex method described in this section maintains dual feasibility and complementary slackness throughout the computations, and it will terminate once primal feasibility has been reached.

The obvious drawback of such a procedure is that while the primal simplex method can be interrupted at any point in time (at least in phase 2) with a primal feasible solution, i.e., a solution that, while not optimal, can actually be implemented, this is not possible with the dual simplex method. This technique has to be fully completed before a feasible (and optimal) solution becomes available.

On the other hand, the dual simplex method allows the user to solve parts of the model and then add variables as they become available.

In order to describe the method, assume again that the original problem in canonical form is described as

$$\begin{aligned} \text{P: Max } z &= \mathbf{cx} \\ \text{s.t. } \mathbf{Ax} &\le \mathbf{b} \\ \mathbf{x} &\ge \mathbf{0}. \end{aligned}$$

This formulation does not include slack, excess, or artificial variables and may also, as opposed to the primal simplex method, contain negative right-hand side values. In case an equation $\mathbf{a}_{i\bullet}\mathbf{x} = b_i$ is present, it can be replaced by two inequalities $\mathbf{a}_{i\bullet}\mathbf{x} \le b_i$ and $\mathbf{a}_{i\bullet}\mathbf{x} \ge b_i$, so that now all constraints are of type "≤". In case of multiple equations, it is not necessary to replace each equation by two "opposing" inequalities, but rather change all existing equations to inequalities of one type and add a single inequality of the other type to the system. Details are provided in Chapter 8.2 of this volume. If the objective function $\max \sum_{j=1}^{n} c_j x_j$ satisfies $c_j \le 0 \ \forall\, j = 1,\ldots, n$, only slack variables have to be added to the left-hand sides of all constraints and we are ready to begin with the procedure. If, however, one or more positive objective function coefficients are given, we have to add the redundant constraint $\sum_{j=1}^{n} x_j \le M$ with $M >> 0$ (or, equivalently, $\sum_{j=1}^{n} x_j + S^* = M$ with slack variable $S^* \ge 0$) to the system, which can now be written as:

$$\begin{aligned} \text{P: Max } & z = \sum_{j=1}^{n} c_j x_j \\ \text{s.t. } & \sum_{j=1}^{n} a_{ij} x_j + S_i = b_i,\ i = 1,\ldots,m \\ & \sum_{j=1}^{n} x_j + S^* = M \qquad (*) \\ & x_j \ge 0 \ \forall\, j = 1,\ldots, n \\ & S^*, S_i \ge 0 \ \forall\, i = 1,\ldots, m. \end{aligned}$$

This problem has a dual feasible solution $x_j = \begin{cases} M, \text{ if } c_j = \max\limits_{k=1,\ldots,n} \{c_k\} \\ 0 \text{ otherwise} \end{cases}$, where ties are broken arbitrarily. This can be seen by assigning variables u_i, $i = 1,\ldots, m$

to the first m primal constraints and a dual variable u_0 to the additional constraint. We then obtain the dual formulation

$$\text{P}_\text{D}\text{: Min } z = \sum_{i=1}^{m} u_i b_i + u_0 M$$

$$\text{s.t.} \quad \sum_{i=1}^{m} a_{ij} u_i + u_0 \geq c_j,\ j = 1,\ldots,n$$

$$u_i \quad \geq 0\ \forall\ i = 1,\ldots, m,$$

for which $u_0 = \max\limits_{k=1,\ldots,n} \{c_k\}$ and $u_i = 0\ \forall\ i = 1,\ldots, m$ is the feasible dual counterpart of the above primal solution. If $c_j \leq 0\ \forall\ j = 1,\ldots, n$, then the origin $x_j = 0\ \forall\ j = 1,\ldots, n$ (and its corresponding dual solution $u_i = 0\ \forall\ i = 1,\ldots, m$) represent a dual feasible solution. If this initial solution also happens to be primal feasible, then the current solution is optimal and the procedure terminates. Suppose now that in the present solution, given by its vector of basic variables $\mathbf{x}_B$ and nonbasic variables

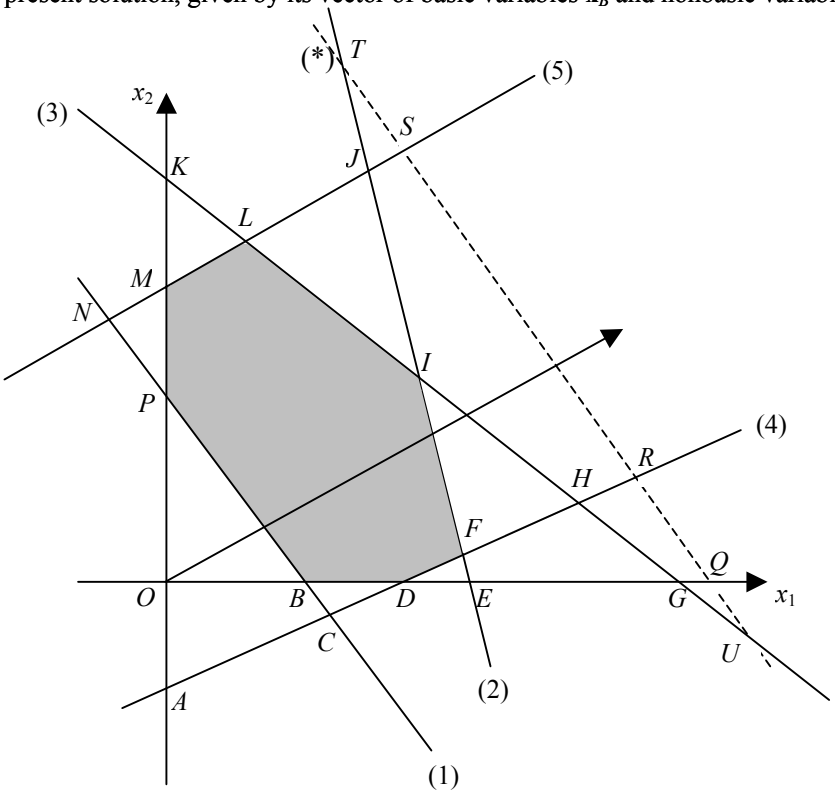


Figure 5.1

$\mathbf{x}_N$, at least one primal constraint is violated. Select one of these violated constraints, say the r-th. (Note that algebraically $S_r < 0$ in the current solution.) A new basic solution is now determined by dropping x_r from the set of basic variables $\mathbf{x}_B$ and introducing a former nonbasic variable, say x_s, into the basis, so that the new basis $\mathbf{x}_B \setminus \{x_r\} \cup \{x_s\}$ is the one among all bases $\mathbf{x}_B \setminus \{x_r\} \cup \{x_j, x_j \in \mathbf{x}_N\}$ for which the value of the objective function decreases by the least amount.

This basis change is repeated until a primal feasible solution has been obtained for the first time; this solution is optimal. Note that in this procedure we still move along the hyperplanes as in the primal algorithm but not necessarily to basic points that are neighbors in the corresponding graph. The above method may be explained by the example in Figure 5.1 where the polytope is the shaded area and the broken line is the additional constraint for which M has been selected reasonably small.

Assuming that a slack S_i has been added to the i-th constraint and S^* belongs to the artificial constraint (*), the Table 5.1 indicates the basic and nonbasic variables as well as all variables with negative values for each of the above basic points $A,\ldots, U$:

Table 5.1

Point	$\mathbf{x_B}$	$\mathbf{x_N}$	$x_j < 0$
O	$S_1, S_2, S_3, S_4, S_5, S^*$	x_1, x_2	S_1
A	$x_2, S_1, S_2, S_3, S_5, S^*$	x_1, S_4	x_2, S_1
B	$x_1, S_2, S_3, S_4, S_5, S^*$	x_2, S_1	none
C	$x_1, x_2, S_2, S_3, S_5, S^*$	S_1, S_4	x_2
D	$x_1, S_1, S_2, S_3, S_5, S^*$	x_2, S_4	none
E	$x_1, S_1, S_3, S_4, S_5, S^*$	x_2, S_2	S_4
F	$x_1, x_2, S_1, S_3, S_5, S^*$	S_2, S_4	none
G	$x_1, S_1, S_2, S_4, S_5, S^*$	x_2, S_3	S_2, S_4
H	$x_1, x_2, S_1, S_2, S_5, S^*$	S_3, S_4	S_2
I	$x_1, x_2, S_1, S_4, S_5, S^*$	S_2, S_3	none
J	$x_1, x_2, S_1, S_3, S_4, S^*$	S_2, S_5	S_3
K	$x_2, S_1, S_2, S_4, S_5, S^*$	x_1, S_3	S_5
L	$x_1, x_2, S_1, S_2, S_4, S^*$	S_3, S_5	none
M	$x_2, S_1, S_2, S_3, S_4, S^*$	x_1, S_5	none
N	$x_1, x_2, S_2, S_3, S_4, S^*$	S_1, S_5	x_1
P	$x_2, S_2, S_3, S_4, S_5, S^*$	x_1, S_1	none
Q	$x_1, S_1, S_2, S_3, S_4, S_5$	x_2, S^*	S_2, S_3, S_4
R	$x_1, x_2, S_1, S_2, S_3, S_5$	S_4, S^*	S_2, S_3
S	$x_1, x_2, S_1, S_2, S_3, S_4$	S_5, S^*	S_2, S_3
T	$x_1, x_2, S_1, S_3, S_4, S_5$	S_2, S^*	S_3, S_5
U	$x_1, x_2, S_1, S_2, S_4, S_5$	S_3, S^*	x_2, S_2, S_4

In the above example c_1, $c_2 > 0$ and $c_1 > c_2$. In the first step, the method moves from the origin O to the point $Q = (M, 0)$, where S_2, S_3 and S_4 are negative. Any one of these three variables can be chosen to leave the basis; here we choose S_4. If S_4 is dropped form the basis, its value is reduced to zero, which means geometrically that the next basic point must lie on hyperplane (4). Moving on a single hyperplane form point Q to hyperplane (4) leads either point R or to point D. Since the move from Q to R reduces the value of the objective function by a much smaller margin than would the move from Q to D, the method will move to point R. (Note that as soon as constraint (4) was selected among the violated constraints, the move to R was mandatory: Moving to D would destroy dual feasibility although in this case primal feasibility would be obtained.) At point R, the second and third constraints are violated, and for our procedure we select (2). Only two moves from R to hyperplane (2) are possible; either to point F or to point T. The reduction in the value of the objective function is smaller if we move to T, and hence this move is made. Note that point S is adjacent to R but was skipped in that move. At point T, constraints (3) and (5) are violated; we select (3). The move from point T to hyperplane (3) leads to either I or U. Point U can not be selected since it would increase the z-value, I is selected. Since point I is also primal feasible, it must be optimal. Various other point sequences of the dual simplex method are possible; the longest is $Q - R - T - J - I$ and the shortest is $Q - T - I$.

We are now able to formally describe the algorithm. The algorithm is initialized with a dual feasible tableau T^0. If this is not initially available, we add the constraint $\sum_{j=1}^{n} x_j + S^* = M$ as the $(m + 1)$-st row in the tableau T^0, select $a_{m+1,s} = 1$ as pivot, where $c_s = \min_{j:c_j<0} \{c_j\}$ and perform a tableau transformation. This step guarantees dual feasibility.

The Dual Simplex Algorithm

Step 1: Is $b_i \geq 0$?
If yes: Go to Step 2.
If no: Go to Step 3.

Step 2: Is the value of the objective function z a function of M?
If yes: Stop, unbounded "optimal" solutions exist.
If no: Stop, the current solution is optimal.

Step 3: Select the r-th row as the pivot row, such that $b_r < 0$.

Step 4: Is there at least one element $a_{rj} < 0, j = 1,\ldots m + n$?
If yes: Go to Step 5.
If no: Stop, the problem has no feasible solution.

Step 5: Select the s-th column as pivot column,

where $\frac{c_s}{|a_{rs}|} = \min\left\{\frac{c_j}{|a_{rj}|} : a_{rj} < 0\right\}$. Perform one tableau transformation with the pivot $a_{rs} < 0$ and go to Step 1.

As opposed to the primal simplex algorithm, the pivot row is selected first and then the pivot column is determined. The same degree of freedom is given for the selection of the pivot row in the dual simplex method as in the primal simplex method for the pivot column; different rules, similar to those described in Section 3.2 can be applied.

In order to illustrate the procedure, consider the following

Example: Let the following linear programming problem be given.

$$\begin{aligned} \text{P: Max } z &= -x_1 + 2x_2 \\ \text{s.t.} \quad & 5x_1 + 4x_2 \geq 20 \\ & x_1 + 5x_2 = 10 \\ & x_1, \quad x_2 \geq 0 \end{aligned}$$

This problem can be written in canonical form as follows:

$$\begin{aligned} \text{P: Max } z &= -x_1 + 2x_2 \\ \text{s.t.} \quad & -5x_1 - 4x_2 \leq -20 \qquad (I) \\ & x_1 + 5x_2 \leq 10 \qquad (II) \\ & -x_1 - 5x_2 \leq -10 \qquad (III) \\ & x_1, \quad x_2 \geq 0 \end{aligned}$$

After adding slacks S_1, S_2 and S_3 and setting up the initial tableau, we notice that the x_2 coefficient in the objective function row is negative. Hence we add the $(m+1)$-st row $x_1 + x_2 + S^* = M$ and obtain the tableau:

T^0:

x_1	x_2	S_1	S_2	S_3	S^*	1	
−5	−4	1	0	0	0	−20	
1	5	0	1	0	0	10	
−1	−5	0	0	1	0	−10	
1	(1)	0	0	0	1	M	←
1	−2	0	0	0	0	0	

In the initialization of the algorithm, the artificial row is used as pivot row and the x_2 row is pivot column since c_2 is the only negative element in the objective function row. After one tableau transformation we obtain a dual feasible basic solution in the tableau

T^1:

x_1	x_2	S_1	S_2	S_3	S^*	1	
−1	0	1	0	0	4	−20+4M	
−4	0	0	1	0	(−5)	10−5M	←
4	0	0	0	1	5	−10+5M	
1	1	0	0	0	1	M	
3	0	0	0	0	2	2M	

Due to the fact the M is very large, the second row is the only row with a negative right hand side; hence it becomes the pivot row. The x_1 and S^* columns are the eligible pivot columns and S^* is selected as entering variable, since $\frac{2}{|-5|} = \min\left\{\frac{3}{|-4|}, \frac{2}{|-5|}\right\}$. After one tableau transformation we obtain

T^2:

x_1	x_2	S_1	S_2	S_3	S^*	1	
$(-\frac{21}{5})$	0	1	$\frac{4}{5}$	0	0	−12	←
$\frac{4}{5}$	0	0	$-\frac{1}{5}$	0	1	M−2	
0	0	0	1	1	0	0	
$\frac{1}{5}$	1	0	$\frac{1}{5}$	0	0	2	
$\frac{7}{5}$	0	0	$\frac{2}{5}$	0	0	4	

The first row is the only pivot eligible row and the x_1 column must be chosen as pivot column. Hence $a_{11} = -\frac{21}{5}$ is the pivot and one iteration results in

T^3:

x_1	x_2	S_1	S_2	S_3	S^*	1
1	0	$-\frac{5}{21}$	$-\frac{4}{21}$	0	0	$\frac{20}{7}$
0	0	$\frac{4}{21}$	$\frac{1}{21}$	0	1	$M-\frac{30}{7}$
0	0	0	1	1	0	0
0	1	$\frac{1}{21}$	$\frac{5}{21}$	0	0	$\frac{10}{7}$
0	0	$\frac{1}{3}$	$\frac{2}{3}$	0	0	0

The solution in tableau T^3 is primal feasible, so that $\overline{\mathbf{x}} = (\frac{20}{7}, \frac{10}{7})^T$ with $\bar{z} = 0$ is optimal.

Note that after the artificial slack variable S^* has left and re-entered the basis in tableau T^2, it will from then on remain in the basis, and therefore its column as well as the row in which it is basic can be deleted. In graphical terms, once the artificial hyperplane that is introduced in the first step, has been left, it is never reached again.

The above procedure can be visualized in Figure 5.2 where the polytope includes all points on the line segment between the points P and Q. Again, the artificial constraint is the broken line for which a sufficiently large value for M was chosen. The solution points generated by the dual simplex method are indicated in the graph by T^0, T^1, T^2 and T^3.

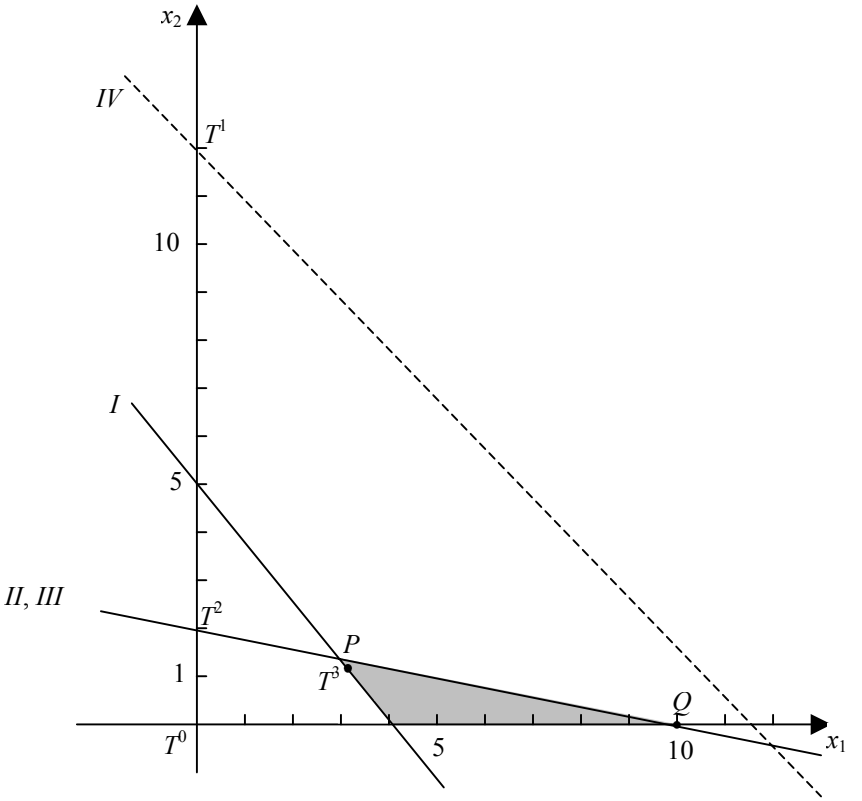


Figure. 5.2

Consider now the four special cases addressed already in Chapter 3. Primal and dual degeneracy in tableaus of the dual simplex method are recognized in exactly the same way as in tableaus of the two phase method.

Unbounded "optimal" solutions exist if a primal feasible solution has been obtained and z is a function of M. As an example, consider the following problem.

$$
\begin{aligned}
\text{P: Max } z = \; & 2x_1 + x_2 \\
\text{s.t.} \quad & -x_1 + x_2 \le 2 \\
& x_1 - x_2 \le 2 \\
& x_1, \; x_2 \ge 0
\end{aligned}
$$

Using the dual simplex method to solve the problem, the (formally) optimal tableau is shown below.

T^{opt}:

x_1	x_2	S_1	S_2	S^*	1
0	0	1	1	0	4
0	1	0	$\frac{1}{2}$	$\frac{1}{2}$	$-1+\frac{1}{2}M$
1	0	0	$\frac{1}{2}$	$\frac{1}{2}$	$1+\frac{1}{2}M$
0	0	0	$\frac{1}{2}$	$\frac{3}{2}$	$-1+\frac{3}{2}M$

By increasing the value of M as much as desired, the value of the objective function also increases as much as desired without destroying primal feasibility.

No feasible solution exists in the dual simplex method, if a pivot row but no pivot column can be found. In other words, there exists at least one row in the tableau with $\sum_{j=1}^{m+n} a_{ij}x_j = b_i$ with $a_{ij} \geq 0$, $j = 1,\ldots, m + n$ and $b_i < 0$, where the vector $\mathbf{x} = (x_1,\ldots, x_{m+n})$ is assumed to include all decision and slack variables. Since $x_j \geq 0 \; \forall \; j = 1,..., m + n$; this constraint clearly cannot be fulfilled. Note that the existence of one such row justifies the immediate abortion of the problem, even if other eligible pivot rows exist, in which negative pivots can be found. Note the similarity of this argument with that applied to the case of unbounded "optimal" solutions when the primal two phase method is applied.

This case may be illustrated by the following

Example: Consider the linear programming problem

$$\begin{aligned} \text{P: Max } z &= 2x_1 + x_2 \\ \text{s.t.} \quad x_2 &\geq 3 \\ x_1 + x_2 &\geq 2 \\ x_1 + x_2 &\leq 1 \\ x_1, \; x_2 &\geq 0 \end{aligned}$$

with a tableau representing the initial dual feasible solution

T^1:

x_1	x_2	S_1	S_2	S_3	1
0	−1	1	0	0	−3
−1	◯	0	1	0	−2
1	1	0	0	1	1
2	1	0	0	0	0

After one tableau transformation, we obtain

T^1:

x_1	x_2	S_1	S_2	S_3	1
1	0	1	−1	0	−1
1	1	0	−1	0	2
0	0	0	1	1	−1
1	0	0	1	0	−2

Although in the pivot-eligible first row, a pivot element, namely $a_{14} < 0$ exists, the third row is also pivot-eligible and no pivot element exists; hence the procedure terminates with the conclusion that the problem has no feasible solution.

In addition to the dual simplex method described above, many other simplex-based methods exist. Most prominently, there are the primal dual simplex method first described by Dantzig *et al*. (1956), the popular revised simplex method with the added and space-saving feature of the product form of the inverse, Ziont's (1969) criss-cross method and many others. They are of mostly technical interest and we refer reader to the appropriate literature, e.g., Eiselt *et al*. (1987), Rardin (1998). Sierksma (2002), and Dantzig and Thapa (1997, 2003).

5.2 The Upper Bounding Technique

In most linear programming problems, many of the variables are bounded from below or from above. Such additional restrictions are also referred to as *secondary constraints*, as opposed to the "regular," or *primary, constraints*. For instance, lower bounds on variables may indicate that production should be at least some minimal level and/or that it should not violate some capacity output.

Clearly, lower and upper bounding constraints can be included in a linear programming as any other constraints. However, upper and lower bounding constraints do not have to be included explicitly in the tableau. Instead, they can be considered implicitly without increasing the size of the simplex tableau. This chapter will concentrate on upper bounding techniques, as nonzero lower bounds on variables can be dealt with by a simple variable transformation shown in Chapter 8. We would also like to point out that much more elaborate schemes exist for dealing with what are known as *generalized upper bounding* techniques that were originally suggested by Dantzig and Van Slyke (1967) and elaborated upon by Cooper and Steinberg (1974). Summaries can be found in Eiselt *et al*. (1987) and Dantzig and Thapa (2003).

The upper bounding technique described here is based on work by Charnes and Lemke (1954) and Dantzig (1954). To facilitate the discussion, assume that each variable x_j has an upper bound u_j assigned to it, where $u_j := \infty$ if no explicit upper bound is known for a variable. The linear programming problem under consideration is then

$$\begin{aligned} \text{P: Max } z &= \mathbf{cx} \\ \text{s.t.} \quad \mathbf{Ax} &\le \mathbf{b} \\ \mathbf{x} &\le \mathbf{u} \\ \mathbf{x} &\ge \mathbf{0}. \end{aligned}$$

Suppose now that at some point during the calculations there is an indicator $c_s < 0$ (if that does not exist, the tableau would represent an optimal solution and we are done). We now choose the s-th column as the pivot column and have to ensure that the entering variable x_s does not exceed its upper bound and that all other basic variables assume values between zero and their respective upper bounds. To facilitate our discussion, all variables and parameters in the present tableau are shown without, while those in the next tableau are shown with an asterisk. Suppose now that the variable x_k is presently in the basis in the i-th row of the current tableau, i.e., $x_k = b_i$. We can then define

$$q^+ := \min_i \left\{ \frac{b_i}{a_{is}} : a_{is} > 0 \right\}, \text{ where } q^+ := +\infty, \text{ if } a_{is} \le 0 \ \forall\ i,$$

$$q^- := \min_i \left\{ \frac{u_k - b_i}{-a_{is}} : a_{is} < 0 \right\}, \text{ where } q^- := +\infty \text{ if } a_{is} \ge 0 \ \forall\ i, \text{ and}$$

$$\varepsilon := \min \{u_s, q^+, q^-\}.$$

Note that u_s, q^+, $q^- \ge 0$, which implies that $\varepsilon \ge 0$. We now have to consider four cases.

Case 1: $\varepsilon = +\infty$, i.e., $u_s = q^+ = q^- = +\infty$. As $u_s = +\infty$, the entering basic variable does not have a finite upper bound, and as $q^+ = q^- = +\infty$, we can conclude that $a_{is} = 0 \ \forall\ i$. Therefore, increasing the value of the entering variable as much as desired will not affect the other basic variables, so that unbounded "optimal" solutions exist.

Case 2: $\varepsilon = u_s < +\infty$. Define x'_s as the slack in the upper bound constraint $x_s \le u_s$, so that $x'_s := u_s - x_s$, or, equivalently, $x_s = u_s - x'_s$. The solution in the current tableau can then be expressed as $\sum_j \mathbf{a}_{\bullet j} x_j = \mathbf{b}$ or $\mathbf{a}_{\bullet s} x_s + \sum_{j \ne s} \mathbf{a}_{\bullet j} x_j = \mathbf{b}$, such that

$$\mathbf{a}_{\bullet s}(u_s - x'_s) + \sum_{j \ne s} \mathbf{a}_{\bullet j} x_j = \mathbf{b}, \text{ or}$$

$$-\mathbf{a}_{\bullet s} x'_s + \sum_{j \ne s} \mathbf{a}_{\bullet j} x_j = \mathbf{b} - \mathbf{a}_{\bullet s} u_s .$$

The above transformation increases the value of x_s to its upper bound, while leaving the basis formally unchanged. We will now show that all new right-hand side values $b_i^* = b_i - a_{is} u_s$, $i = 1, \ldots, m$, are nonnegative and that none of these

values exceeds its respective upper bound. Recall that $x_k = b_i$, i.e., x_k is basic in the i-th row. Then there are three possibilities: the parameter a_{is} is either positive, negative, or zero. We consider each possibility in turn.

(i) $a_{is} > 0$. Since $u_s = \varepsilon \le q^+ \le \dfrac{b_i}{a_{is}}$, it follows that

$$b_i^* = b_i - a_{is}u_s \ge b_i - a_{is}\frac{b_i}{a_{is}} = 0 \text{ and that}$$

$$b_i^* = b_i - a_{is}u_s \le b_i \le u_k, \text{ and hence } 0 \le b_i^* \le u_k.$$

(ii) $a_{is} < 0$. Since $u_s = \varepsilon \le q^- \le \dfrac{u_k - b_i}{-a_{is}}$, it follows that

$$b_i^* = \mathrm{b_i} - a_{is}u_s \le b_i - \mathrm{a_{is}}\frac{u_k - b_i}{-a_{is}} = u_k \text{ and that}$$

$$b_i^* = \mathrm{b_i} - a_{is}u_s \ge b_i \ge 0, \text{ and hence } 0 \le b_i^* \le u_k.$$

(iii) $a_{is} = 0$. Then $b_i^* = b_i - a_{is}u_s = b_i \in [0, u_k]$ and hence $0 \le b_i^* \le u_k$.

Due to the fact that $c_s < 0$, the new objective function value is $z^* = z - c_s u_s \ge z$ (and $z^* > z$ for $u_s > 0$).

Case 3: $\varepsilon = q^+ < +\infty$. The case corresponds to the regular primal simplex method. Assume that the minimum ratio is obtained for $i = r$, i.e., $q^+ = \dfrac{b_r}{a_{rs}} \le \dfrac{b_i}{a_{is}} \quad \forall\, i$ with $a_{is} > 0$, so that a_{rs} is the pivot element. We now perform a regular tableau transformation. The new tableau will then contain a basic solution and its r-th right-hand side value is $b_r^* = \dfrac{b_r}{a_{rs}} \ge 0$. On the other hand, $b_r^* = \dfrac{b_r}{a_{rs}} = q^+ = \varepsilon \le u_s$, so that $0 \le b_r^* \le u_s$. Consider now the i-th row, $i \ne r$, in which x_k is basic. As in Case 2, there are three possibilities.

(i) $a_{is} > 0$. In this case, $b_i^* = b_i - a_{is}\dfrac{b_r}{a_{rs}} \le b_i \le u_k$, since a_{is}, b_r, and a_{rs} are all nonnegative. Furthermore, since $\dfrac{b_r}{a_{rs}} = q^+ \le \dfrac{b_i}{a_{is}}$, it follows that

$$b_i^* = b_i - a_{is}\frac{b_r}{a_{rs}} \ge b_i - a_{is}\frac{b_i}{a_{is}} = 0, \text{ and hence } 0 \le b_i^* \le u_k.$$

(*ii*) $a_{is} < 0$. In this case, $b_i^* = b_i - a_{is}\dfrac{b_r}{a_{rs}} \geq b_i \geq 0$, since $a_{is} < 0$, $a_{rs} > 0$, and $b_r \geq 0$. Due to the fact that $\dfrac{b_r}{a_{rs}} = q^+ = \varepsilon \leq q^- \leq \dfrac{u_k - b_i}{-a_{is}}$, we obtain $b_i^* = b_i - a_{is}\dfrac{b_r}{a_{rs}} \leq b_i - a_{is}\dfrac{u_k - b_i}{-a_{is}} = u_k$, and hence $0 \leq b_i^* \leq u_k$.

(*iii*) $a_{is} = 0$. Then $b_i^* = b_i - a_{is}\dfrac{b_r}{a_{rs}} = b_i \in [0, u_k]$ and hence $0 \leq b_i^* \leq u_k$.

Since $c_s < 0$, we can conclude that $z^* = z - c_s\dfrac{b_r}{a_{rs}} \geq z$ (and $z^* > z$ if $b_r > 0$, i.e., in the case of primal nondegeneracy).

Case 4: $\varepsilon = q^- < +\infty$. Assume that the minimum ratio is obtained for $i = \nu$ and that x_μ is basic in the ν-th row, i.e., $q^- = \dfrac{u_\mu - b_\nu}{-a_{\nu s}} \leq \dfrac{u_k - b_i}{-a_{is}}\ \forall\ i : a_{is} < 0$ and $x_\mu = b_\nu$. Set now $x'_\mu = u_\mu - x_\mu$, so that $x_\mu = u_\mu - x'_\mu$. The solution in the current tableau can then be expressed as $\sum_j \mathbf{a}_{\bullet j} x_j = \mathbf{b}$ or $\mathbf{a}_{\bullet\mu} x_\mu + \sum_{j\neq\mu} \mathbf{a}_{\bullet j} x_j = \mathbf{b}$, so that

$$\mathbf{a}_{\bullet\mu}(u_\mu - x'_\mu) + \sum_{j\neq\mu} \mathbf{a}_{\bullet j} x_j = \mathbf{b} \text{ or } -\mathbf{a}_{\bullet\mu} x'_\mu + \sum_{j\neq\mu} \mathbf{a}_{\bullet j} x_j = \mathbf{b} - \mathbf{a}_{\bullet\mu} u_\mu .$$

Since x_μ was in the basis in the ν-th row, it follows that $a_{\nu\mu} = 1$. After replacing x_μ by x'_μ, we find that the column that belongs to x'_μ is $-\mathbf{e}_\nu$, i.e., there is presently no basic variable in the ν-th row. Now $a_{\nu s} < 0$ is selected as pivot, so that x_s enters the basis and x_ν leaves, so that the new tableau still contains a full basis. Consider now the ν-th row, in which x_s is basic. Its right-hand side value is $b_\nu^* = \dfrac{b_\nu - a_{\nu\mu}u_\mu}{a_{\nu s}} = \dfrac{b_\nu - u_\mu}{a_{\nu s}} = q^- = \varepsilon \leq u_s$. On the other hand, $b_\nu^* = \dfrac{b_\nu - u_\mu}{a_{\nu s}} = \dfrac{x_\mu - u_\mu}{a_{\nu s}} \geq 0$, so that $0 \leq b_\nu^* \leq u_s$. Consider now the i-th row with $i \neq \nu$, in which x_k is basic. As before, there are three possibilities that are treated separately.

(*i*) $a_{is} > 0$. Since $\dfrac{u_\mu - b_\nu}{-a_{\nu s}} = q^- = \varepsilon \leq q^+ \leq \dfrac{b_i}{a_{is}}$, it follows that $b_i^* = b_i - a_{is}\dfrac{b_\nu - u_\mu}{a_{\nu s}} \geq b_i - a_{is}\dfrac{b_i}{a_{is}} \geq 0$ and $b_i^* = b_i - a_{is}q^- \leq b_i \leq u_k$, hence $0 \leq b_i^* \leq u_k$.

(*ii*) $a_{is} < 0$. Since $b_i^* = b_i - a_{is}q^- \geq b_i \geq 0$ and as $\frac{u_\mu - b_\nu}{-a_{\nu s}} \leq \frac{u_k - b_i}{-a_{is}}$, we obtain $b_i^* = b_i - a_{is}\frac{b_\nu - u_\mu}{a_{\nu s}} \leq b_i - a_{is}\frac{b_i - u_k}{a_{is}} \leq u_k$, and hence $0 \leq b_i^* \leq u_k$.

(*iii*) $a_{is} = 0$. Now $b_i^* = b_i - a_{is}q^- = b_i \in [0, u_k]$, and hence $0 \leq b_i^* \leq u_k$.. Since $c_s < 0$, we can conclude that the new value of the objective function is $z^* = z - c_s\frac{b_\nu - u_\mu}{a_{\nu s}} = z - c_s q^- \geq z$ (and $z^* > z$ for $b_\nu < u_\mu$).

Now all possible cases have been examined and the solution procedure can be stated in algorithmic form. Only those steps that relate directly the upper bounding technique will be described, all other steps are carried out according to the primal simplex method. We also assume that all lower bounds have been transformed to zero and that the *s*-th column has been chosen as the pivot column, so that $c_s < 0$.

The Upper Bounding Technique

Step 1: Define

$$q^+ := \min_i \left\{ \frac{b_i}{a_{is}} : a_{is} > 0 \right\}, \text{ where } q^+ := +\infty, \text{ if } a_{is} \leq 0 \ \forall\ i,$$

$$q^- := \min_i \left\{ \frac{u_k - b_i}{-a_{is}} : a_{is} < 0 \right\}, \text{ where } x_k = b_i \text{ (i.e., } x_k \text{ is basic in row } i\text{), and}$$

where

$$q^- := +\infty \text{ if } a_{is} \geq 0 \ \forall\ i, \text{ and } \varepsilon := \min\ \{u_s, q^+, q^-\}.$$

Step 2: Is $\varepsilon = +\infty$?

If yes: Stop, unbounded "optimal" solutions exist (Case 1).

If no: Go to Step 3.

Step 3: Is $\varepsilon = u_s$?

If yes: Set $x_s' = u_s - x_s$ and proceed with the primal simplex method.

If no: Go to Step 4.

Step 4: Is $\varepsilon = q^+$?

If yes: Assuming that $q^+ = \frac{b_r}{a_{rs}}$, choose a_{rs} as the pivot element, perform a tableau transformation, and proceed with the primal simplex method.

If no: Go to Step 5.

Step 5: Assume that $q^- = \dfrac{u_\mu - b_\nu}{-a_{\nu s}}$, where $x_\mu = b_\nu$ (i.e., x_μ is basic in row ν). Set $x'_\mu = u_\mu - x_\mu$. choose $a_{\nu s}$ as the pivot element, perform a tableau transformation, and proceed with the primal simplex method.

In order to illustrate the upper bounding technique, consider the following

Example: Let the following linear programming problem be given:

P: Max $z = 3x_1 + 2x_2 + 9$

$$\begin{array}{lrl} \text{s.t.} & 15x_1 + 5x_2 & \le 40 \\ & 6x_1 + 6x_2 & \le 18 \\ & 10x_1 + 5x_2 & \le 25\tfrac{1}{2} \\ & x_1 & \le 2 \\ & x_2 & \le 2\tfrac{1}{2} \\ & x_1, \quad x_2 & \ge 0. \end{array}$$

The initial tableau is shown in

T^1:

x_1	x_2	S_1	S_2	S_3	1
15	5	1	0	0	40
6	6	0	1	0	18
10	5	0	0	1	25½
–3	–2	0	0	0	9

For expository reasons we choose the x_2 column as the pivot column. Now $u_2 = 2\tfrac{1}{2}$, $q^+ = \min\{\frac{40}{5}, \frac{18}{6}, \frac{25\frac{1}{2}}{5}\}$, $q^- = \infty$, so that $\varepsilon = \min\{2\tfrac{1}{2}, 3, \infty\} = 2\tfrac{1}{2}$. As $\varepsilon = u_2$ (Step 3 in the algorithm), we perform the variable transformation $x'_2 = u_2 - x_2 = 2\tfrac{1}{2} - x_2$, and obtain the second tableau

T^2:

x_1	x'_2	S_1	S_2	S_3	1
15	–5	1	0	0	27½
(6)	–6	0	1	0	3
10	–5	0	0	1	13
–3	2	0	0	0	14

Now $u_1 = 2$, $q^+ = \min\{\frac{27\frac{1}{2}}{15}, \frac{3}{6}, \frac{13}{10}\} = \tfrac{1}{2}$, $q^- = \infty$, and hence $\varepsilon = \{2, \tfrac{1}{2}, \infty\} = \tfrac{1}{2}$. As q^+ determines ε, a regular simplex step will be carried out. Pivoting is performed on the circled element in the tableau T^2, resulting in tableau

T^3:

x_1	x_2'	S_1	S_2	S_3	1
0	10	1	$-\frac{5}{2}$	0	20
1	−1	0	$\frac{1}{6}$	0	$\frac{1}{2}$
0	5	0	$-\frac{5}{3}$	1	8
0	−1	0	$\frac{1}{2}$	0	15½

At this point, $u_2 = 2½$, $q^+ = \min\left\{\frac{20}{10}, \frac{8}{5}\right\} = \frac{8}{5}$, $q^- = \min\left\{\frac{2-½}{-(-1))}\right\} = 1½$, thus $\varepsilon = \min\left\{2\frac{1}{2}, 1\frac{3}{5}, 1\frac{1}{2}\right\} = 1½$. Since ε is determined by q^-, we first have to make an upper bound substitution for x_1, so that $x_1' = u_1 - x_1$, i.e., $x_1' = 2 - x_1$. As a result we obtain the incomplete tableau

T^4:

x_1'	x_2'	S_1	S_2	S_3	1
0	10	1	$-\frac{5}{2}$	0	20
−1	(−1)	0	$\frac{1}{6}$	0	$-\frac{3}{2}$
0	5	0	$-\frac{5}{3}$	1	8
0	−1	0	$\frac{1}{2}$	0	15½

After one standard simplex iteration with the pivot circled in the tableau T^4 we obtain tableau

T^5:

x_1'	x_2'	S_1	S_2	S_3	1
−10	0	1	$-\frac{5}{6}$	0	5
1	1	0	$-\frac{1}{6}$	0	1½
−5	0	0	$-\frac{5}{6}$	1	½
1	0	0	$\frac{1}{3}$	0	17

It is apparent that the tableau T^5 satisfies the criterion for optimality. However, since its solution is expressed in terms of the variables x_1' and x_2', we have to recreate the values of the original variables, which results in $\bar{x}_1 = 2 - x_1' = 2$ and $\bar{x}_2 = 2½ - x_2' = 1$, so that $\bar{z} = 17$.

5.3 Column Generation

Many practical problems lead to linear programming models of a large scale with thousands or even tens of thousands of rows. Worse, the number of columns (i.e., variables) may easily reach millions or even billions, as we will demonstrate in the examples below. Invariably, larger scale problems possess special structures and have coefficient matrices with a sparsity of only a few percent, often just a fraction of a percent. The revised simplex method is then preferable to the regular simplex methods since it has lower storage requirements and is computationally more efficient. However, for really large-scale problems, this is not sufficient and more specialized techniques are needed.

Considering the computational capability at the time, the decomposition method by Dantzig and Wolfe (1960, 1961) held the promise of being able to solve quite large linear programming problems quickly and efficiently. But the method turned out to be less successful numerically than had first been hoped, while at the same time the simplex method had been improved (e.g., by virtue of the inverse in product form, *LU* decomposition, etc.) and the major linear programming software packages refined to such an extent that the Dantzig-Wolfe decomposition method was never really able to compete. The decomposition or partitioning method of Benders (1962), which can be seen as a dual to the Dantzig-Wolfe method, met with the same fate. Although interesting in themselves due to their theoretical properties and economic interpretation, we will not describe these methods here and instead refer to the original contributions or Eiselt *et al*. (1987) for a full treatment.

A method that has withstood the test of time and which appears to be the only workable strategy for very large scale problems with millions or even billions of columns is the *column-generation* technique which we describe below.

Consider a linear programming problem in canonical form as

$$\begin{aligned} \text{P: Max } z &= \mathbf{cx} \\ \text{s.t.} \quad \mathbf{Ax} &\le \mathbf{b} \\ \mathbf{x} &\ge \mathbf{0}, \end{aligned}$$

where the number m of rows is moderately large, while the number n of columns may be astronomical. Assume now that $k > m$ (for some $k << n$) of the columns of the matrix $\mathbf{A}$ have "somehow" been generated or are already known, forming the $[m \times k]$-dimensional submatrix $\mathbf{A}^k$ of $\mathbf{A}$. We denote the corresponding subvectors of $\mathbf{c}$ and $\mathbf{x}$ by $\mathbf{c}^k$ and $\mathbf{x}^k$, respectively. We can then solve the simplified problem

$$\begin{aligned} \text{P}^k\text{:} \ \max_{\mathbf{x}^k} z^k &= \mathbf{c}^k\mathbf{x}^k \\ \text{s.t.} \quad \mathbf{A}^k\mathbf{x}^k &\le \mathbf{b} \\ \mathbf{x}^k &\ge \mathbf{0}, \end{aligned}$$

i.e., problem P restricted to the k columns under consideration. The result will be a basic (feasible and) optimal solution $\bar{\mathbf{x}}^k$. Note that at most m of the components of $\bar{\mathbf{x}}^k$ are nonzero and that $\bar{\mathbf{x}}^k$ will yield a feasible, but potentially nonoptimal, solution to the original problem P. Let now the m-dimensional row vector $\bar{\mathbf{u}}^k$ denote an optimal solution to the dual of P^k; this can be found in the optimal simplex tableau for P^k. Since $\bar{\mathbf{x}}^k$ is optimal for P^k, we know that the reduced cost coefficients $c_j^k - \bar{\mathbf{u}}^k \mathbf{a}_{\bullet j}^k \geq 0$, where $\mathbf{a}_{\bullet j}^k$ denotes the j-th column vector of $\mathbf{A}^k$. If in addition, $c_j - \bar{\mathbf{u}}^k \mathbf{a}_{\bullet j} \geq 0$ for all columns j in the original matrix $\mathbf{A}$, then $\bar{\mathbf{x}}^k$ constitutes an optimal solution to the original problem P as well. The basic idea of column generation is now to consider the subproblem

$$P_k^{SUB}: \ \operatorname{Min}_j z_k^{SUB} = c_j - \bar{\mathbf{u}}^k \mathbf{a}_{\bullet j}$$

where j ranges over all columns of the original problem P. If the optimal objective function value $\bar{z}_k^{SUB}$ of this subproblem is nonnegative, an optimal solution for P has been found, otherwise $\bar{z}_k^{SUB} < 0$ and the corresponding variable x_j can be brought into the basis of the simplified problem P^{k+1}, having the variable x_j in addition to the variables of P^k. The procedure then continues with P^{k+1} replacing P^k and so forth, until an optimal solution to the original problem P has been found. It is the generation of columns occurring when solving the problem P_k^{SUB} that has given the procedure its name; a variety of techniques may be applied to $\bar{z}_k^{SUB}$, depending on the particular problem at hand.

We will now describe the column generation procedure in algorithmic form. The special technique for solving the subproblem P_k^{SUB} will vary, depending on the particular application at hand. We initialize the method having selected $k > m$ out of the $n >> k$ columns of the matrix $\mathbf{A}$.

The Column Generation Technique

Step 1: Solve the simplified problem

$$\begin{aligned} P^k: \ \operatorname{Max}_{x^k} \ & z^k = \mathbf{c}^k \mathbf{x}^k \\ \text{s.t.} \quad & \mathbf{A}^k \mathbf{x}^k \leq \mathbf{b} \\ & \mathbf{x}^k \geq \mathbf{0}, \end{aligned}$$

from which we obtain the optimal solution $\bar{\mathbf{x}}^k, \bar{\mathbf{u}}^k$.

Step 2: Solve the subproblem

$$\mathrm{P}_k^{SUB}: \underset{j=1,\ldots,n}{\mathrm{Min}}\ z_k^{SUB} = c_j - \bar{\mathbf{u}}^k \mathbf{a}_{\bullet j},$$

from which we obtain an optimal solution $\bar{j}$ with an optimal objective function value $\bar{z}_k^{SUB}$.

Step 3: Is $\bar{z}_k^{SUB} \geq 0$?

If yes: Stop, $\bar{\mathbf{x}} := \bar{\mathbf{x}}^k$ is an optimal solution to the original problem P.

If no: Form an $[m \times (k+1)]$-dimensional matrix $\mathbf{A}^{k+1}$ by augmenting the $[m \times k]$-dimensional matrix $\mathbf{A}^k$ with the row column $\mathbf{a}_{\bullet \bar{j}}$. Set $k := k+1$ and go to Step 1.

We will illustrate the technique by solving a simple example of the celebrated cutting stock problem that was first described by Gilmore and Gomory (1961, 1963). Detailed accounts are provided in Chapter 2 of this volume, further details are can be found in Sierksma (2002) and Eiselt *et al*. (1987).

Example: Assume that a company produces and sells metal rods of three different lengths, *viz*., 3, 4, and 5 feet. These rods are obtained by cutting 15-foot rods which the company has in stock. The cutting can be accomplished by cutting according to nine different cutting patterns as shown in Table 5.2. These patterns have been chosen according to some rule, e.g., their property not to produce inordinate amounts of waste. The table indicates the number of rods produced by a pattern.

Table 5.2

Pattern number	1	2	3	4	5	6	7	8	9
3-foot rods in pattern	5	3	3	2	2	1	1	0	0
4-foot rods in pattern	0	1	0	2	1	3	0	2	0
5-foot rods in pattern	0	0	1	0	1	0	2	1	3

In real life, the number of patterns will be very large (millions of numbers of patterns even for modest numbers of different rod sizes), especially as the size of the raw material gets large. Assume now that the customers demand twenty 3 ft rods, thirty 4 ft rods, and ten 5 ft rods. The objective is now to cut the existing rods so as to satisfy customer demand while, at the same time, cut as few existing rods as possible.

Denoting by x_j the number of 15 ft rods that will be cut according to pattern j, j = 1, …, 9, the problem can then be formulated as

$$\begin{aligned}
\text{P: Min } z = {} & x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 \\
\text{s.t.} \quad & 5x_1 + 3x_2 + 3x_3 + 2x_4 + 2x_5 + x_6 + x_7 \geq 20 \\
& x_2 + 2x_4 + x_5 + 3x_6 + 2x_8 \geq 30 \\
& x_3 + x_5 + 2x_7 + x_8 + 3x_9 \geq 10 \\
& x_j \geq 0 \ \forall\, j = 1, \ldots, 9,
\end{aligned}$$

where the objective function minimizes the usage of 15 ft. rods, and the three structural constraints express the demand for 3 ft, 4 ft, and 5 ft rods.

Due to the nature of the problem, the variables x_j need to be integer, a requirement that we will ignore in order to simplify our discussion. In their original contribution, Gilmore and Gomory deal with the cutting of paper rolls and their contribution includes a discussion concerning how fractional rolls could be cut. Here, we simply round our variables up to the nearest integer.

Assume now that the patterns 1, 2, 3, and 4 have been chosen to be included in the solution, so that $k = 4$. The simplified problem is then

$$\begin{aligned}
\text{P}^4\text{: Min } z^4 = {} & x_1 + x_2 + x_3 + x_4 \\
\text{s.t.} \quad & 5x_1 + 3x_2 + 3x_3 + 2x_4 \geq 20 \\
& x_2 + 2x_4 \geq 30 \\
& x_3 \geq 10 \\
& x_1, \quad x_2, \quad x_3, \quad x_4 \geq 0.
\end{aligned}$$

Solving P^4 results in the optimal solution $\bar{x}_1^4 = \bar{x}_2^4 = 0$, $\bar{x}_3^4 = 10$, and $\bar{x}_4^4 = 15$, and the optimal dual solution is $\bar{u}_1^4 = 0$, $\bar{u}_2^4 = ½$, $\bar{u}_3^4 = 1$, with $\bar{z}^4 = 25$, so that twenty-five 15 ft rods need to be cut, ten by using pattern 3 and fifteen by using pattern 4.

The subproblem is then

$$\text{P}^{\text{SUB}}\text{:} \ \min_{j=1,\ldots,9} z^{SUB} = 1 - \bar{\mathbf{u}}^4 \mathbf{a}_{\bullet j}.$$

It is apparent that the only requirement of a cutting pattern is that the total length of the rods in the pattern cannot exceed 15 ft. Hence, any column $\mathbf{a}_\bullet = [a_1, a_2, a_3]$ must therefore satisfy $3a_1 + 4a_2 + 5a_3 \leq 15$ (where a_1 denotes the number of 3 ft lengths cut from the available 15 ft rod, and similar for a_2 and a_3), so that the subproblem can be reformulated as

$$\mathrm{P}_{\mathbf{a}}^{\mathrm{SUB}}: \underset{a_1,a_2,a_3}{\text{Min}}\ z^{SUB} = 1 - \tfrac{1}{2}a_2 - a_3$$
$$\text{s.t.}\quad 3a_1 + 4a_2 + 5a_3 \le 15$$
$$a_1,\quad a_2,\quad a_3 \ge 0 \text{ and integer.}$$

The problem $\mathrm{P}_{\mathbf{a}}^{\mathrm{SUB}}$ is a so-called knapsack problem, i.e., an integer programming problem with a single constraint, that has positive coefficients in the constraint. Here, the unique solution to the problem $\mathrm{P}_{\mathbf{a}}^{\mathrm{SUB}}$ is $[\bar{a}_1, \bar{a}_2, \bar{a}_3] = [0, 0, 3]$, so that $\bar{z}^{SUB} = -2 < 0$, generating the corresponding column $\mathbf{a}_{\bullet 9} = [0, 0, 3]^T$. Hence, the new problem P^5 is then

$$\mathrm{P}^5: \text{Min } z = x_1 + x_2 + x_3 + x_4 + x_9$$
$$\begin{aligned} \text{s.t.}\quad & 5x_1 + 3x_2 + 3x_3 + 2x_4 \ge 20 \\ & x_2 + 2x_4 \ge 30 \\ & x_3 + 3x_9 \ge 10 \\ & x_1,\ x_2,\ x_3,\ x_4,\ x_9 \ge 0. \end{aligned}$$

The solution to P^5 is $\bar{x}_1^5 = \bar{x}_2^5 = \bar{x}_3^5 = 0$, $\bar{x}_4^5 = 15$, and $\bar{x}_9^5 = \frac{10}{3}$, while the dual solution is $\bar{u}_1^5 = 0, \bar{u}_2^5 = \tfrac{1}{2}$, and $\bar{u}_3^5 = \tfrac{1}{3}$, so that $\bar{z}^5 = 18\tfrac{1}{3}$ and nineteen 15-foot rods need to be cut; fifteen by using pattern 4 and four by using pattern 9.

The subproblem is now

$$\mathrm{P}_{\mathbf{a}}^{\mathrm{SUB}}: \underset{a_1,a_2,a_3}{\text{Min}}\ \mathrm{z}^{\mathrm{SUB}} = 1 - \tfrac{1}{2}a_2 - \tfrac{1}{3}a_3$$
$$\text{s.t.}\quad 3a_1 + 4a_2 + 5a_3 \le 15$$
$$a_1,\quad a_2,\quad a_3 \ge 0 \text{ and integer}$$

and an optimal solution is $[\bar{a}_1, \bar{a}_2, \bar{a}_3] = [1, 3, 0]$ with $\bar{z}^{\mathrm{SUB}} = -\tfrac{1}{2} < 0$. The corresponding column generated is therefore $\mathbf{a}_{\bullet 6} = [1, 3, 0]^T$, so that the new problem P^6 is

$$\mathrm{P}^6: \text{Min } z = x_1 + x_2 + x_3 + x_4 + x_6 + x_9$$
$$\begin{aligned} \text{s.t.}\quad & 5x_1 + 3x_2 + 3x_3 + 2x_4 + x_6 \ge 20 \\ & x_2 + 2x_4 + 3x_6 \ge 30 \\ & x_3 + 3x_9 \ge 10 \\ & x_1,\ x_2,\ x_3,\ x_4,\ x_6,\ x_9 \ge 0. \end{aligned}$$

The solution to the problem P^6 is $\bar{x}_1^6 = 2$, $\bar{x}_2^6 = \bar{x}_3^6 = \bar{x}_4^6 = 0$, $\bar{x}_6^6 = 10$, $\bar{x}_9^6 = \frac{10}{3}$ with $\bar{z}^6 = 15\frac{1}{3}$. The dual solution is $\bar{u}_1^6 = \frac{1}{5}$, $\bar{u}_2^6 = \frac{4}{15}$, $\bar{u}_3^6 = \frac{1}{3}$, and sixteen 15-foot rods need to be cut, two using pattern 2, ten using pattern 6, and four using pattern 9.

Now the subproblem is

$$\begin{aligned} P_{\mathbf{a}}^{SUB}: \quad & \underset{a_1,a_2,a_3}{\text{Min}} \; z^{SUB} = 1 - \tfrac{1}{5}a_1 - \tfrac{4}{15}a_2 - \tfrac{1}{3}a_3 \\ & \text{s.t.} \quad 3a_1 + 4a_2 + 5a_3 \le 15 \\ & \qquad a_1, \quad a_2, \quad a_3 \ge 0 \text{ and integer} \end{aligned}$$

that has, among others, an optimal solution $[\bar{a}_1, \bar{a}_2, \bar{a}_3]$ = [2, 1, 1], so that $\bar{z}^{SUB}$ = 0, therefore an optimal solution to the original problem has been found. The optimal solution is the one stated above with sixteen 15-foot rods to be cut, two using pattern 2, ten using pattern 3, and four using pattern 9. The total trim loss is then 22 feet of rods (i.e., six 2-ft pieces and ten 1-ft pieces).

The best-known successful application of column generation is in the airline industry, where airplane flight crews are assigned to sequences of flights. Savings in the order of $13 million per year were reported by one major airline when improved schedules were computed; for details, see Anbil *et al*. (1991). In general, column generation procedures have found widespread applications in software for solving large-scale linear programming problems in the area of vehicle routing and scheduling.

6 POSTOPTIMALITY ANALYSES

As discussed in Chapter 1 of this volume, one of the assumptions in linear programming is that all parameters are deterministic, i.e., assumed to be known with certainty. Clearly, this assumption will rarely be satisfied in practice. One popular way to get around the problem, i.e., dealing with uncertainty while keeping the simple structure and efficient solution methods of linear programming, are sensitivity analyses. In essence, sensitivity (sometimes also called "what…if?") analyses deal with the effects of parameter changes on the optimal solution. Typical examples include analyses regarding the effects of price changes on a cutting plan, changes in the demand structure on the allocation of resources, and technological changes (such as faster throughputs in some machines) on a production schedule.

Depending on the magnitude of the changes, some of these questions can be answered on the basis of the information provided in the final tableau (or, similarly, in the summary of results and the sensitivity analyses provided by the solver). This information is, however, only available for relatively minor changes, i.e., changes in the vicinity of the present solution. Major changes will require sensitivity analyses that essentially necessitate to resolve the problem with the new parameters. In case of sensitivity analyses that are based on the information provided in an optimal tableau, it is essential that—with one exception that is clearly labeled below—all analyses investigate the changes of the solution based on the change of a single parameter. For instance, if the demand for a product may be somewhat lower than expected, then we can analyze the change of such a drop in demand, but not a simultaneous change of, say, the price of one of the products.

From a technical point of view, if there are a number of simultaneous changes that are to be investigated and it has been decided to resolve the problem, it may be beneficial to start the new optimization run not with a "cold start," i.e., at the origin as we would with a new and yet unsolved problem, but to use a "warm" or "hot start," defined as an optimization run that commences with the solution that was optimal before the parameter changes were made. Such considerations are, of course, only meaningful for models that are of very large size and take a very long time to solve.

Investigations that deal with the changes in the optimal solution due to variation in the data, are known by the name *postoptimality analysis*. Form the original work of Manne (1953), other important contributions have been made by Gass and Saaty (1955), Dinkelbach (1969), Gal (1979, 1984) and others. As we will see from the discussions in this section, many important postoptimality questions can be fully answered from the numerical information available in the final tableau of the simplex procedure. Implicitly, these techniques make use of the relationships between the primal and the dual as explained in Chapter 4.

Before discussing any of the many aspects of postoptimality analysis, some important points should be made clear. Recall that, if not explicitly stated otherwise, only one change at a time will be considered. The various cases which we will discuss are those which appear in Figure 6.1.

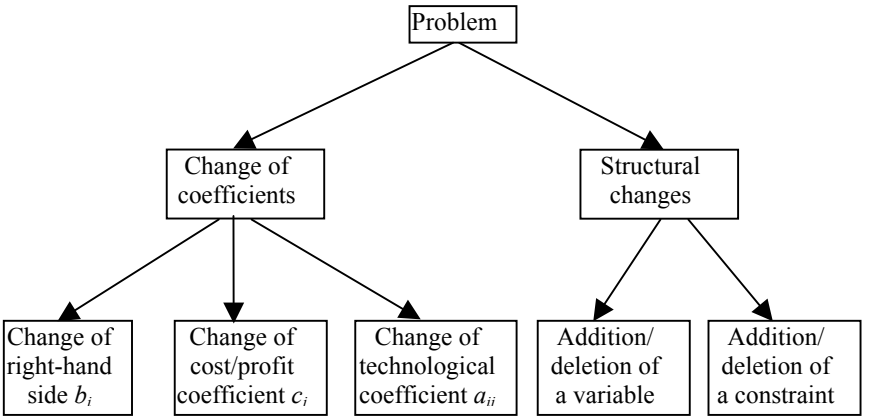


Figure 6.1

Note, however, that things are not as clear cut as Figure 6.1 may suggest. For example, the removal of a constraint (a structural change) is equivalent to changing the right-hand side value of that constraint so as to render it redundant (a coefficient change). Still, the distinction is useful for most practical purposes. Generally speaking, except for maybe the addition or deletion of individual constraints, structural changes are best dealt with by re-solving the problem.

It is also important to distinguish between two types of postoptimality analysis, namely *sensitivity analysis* and *parametric programming*. In sensitivity analysis, the question is: what is the range over which a given parameter can change without changing the optimal basis?, whereas in parametric programming the question is: what happens if an actual change is such that it does not fall within the above range? As an example, consider a production plan that includes a particular item for which the given price was \$5. Then, sensitivity analysis could indicate

that for a price between, say, \$3 and \$6, the optimal basis and the production plan would not change. Furthermore, it will determine what the solution is, given that the new price falls within this interval. Parametric programming, on the other hand, will answer questions such as: what happens if the price for the above product is \$13? What if it is \$1?

Depending on the given parameters, a solution may be sensitive or insensitive to changes. If the addition or subtraction of a few cents to the unit cost or price (or the addition/subtraction of a few units to the right-hand side) does already make the current optimal solution infeasible or not optimal, the model has a very sensitive solution. On the other hand, if substantial changes can be made to the input parameters with only small, if any, changes resulting—such as in the above numerical example—then the solution is said to be insensitive to changes. Such models are frequently referred to as *robust* or sometimes as *stable*.

6.1 Graphical Sensitivity Analysis

In this section we will introduce changes to the parameters b_i and c_j and demonstrate what happens to the optimal solution if the right-hand side values and the coefficients of the objective function increase and decrease.

Throughout this chapter, we will use Δc_j and Δb_i to denote the change of the j-th unit cost/profit coefficient and the i-th right-hand side value, respectively.

First, consider changes of the right-hand side values given a set of constraints as shown in Figure 6.2.

Given the above constraints *I*, *II*, and *III*, where constraint *III* presently has a right-hand side value of b_0, the shaded area indicates the set of feasible solution. Applying the graphical solution technique, it is apparent that point *B* represents the unique optimal solution. The optimal basis at point *B* includes the variables x_1, x_2, and S_1, the slack of constraint *I*.

If the right-hand side b_0 were to be increased to b_1, the hyperplane belonging to this constraint would shift in parallel fashion and the new polytope has the extreme points OAB_1D_1E with B_1 as the new optimal solution. Note that the optimal basis is still the same, namely x_1, x_2, S_1, but the optimal solution has moved form *B* to B_1 so that the basic variables x_1, x_2, and S_1 now have different values. Increasing the right-hand side further to b_2, the polytope is *OACE* and point *C* is now the optimal solution. At this point we encounter primal degeneracy, as the constraints *I*, *II*, and *III* all intersect at point *C*. The basis at this point includes x_1 and x_2 as well as exactly one of the slacks S_1, S_2, and S_3 with a value of zero. Increasing the right-hand side further to b_3 will render constraint *III* redundant, so that the polytope remains *OACE* with the now nondegenerate optimal solution *C* and basis x_1, x_2, S_3.

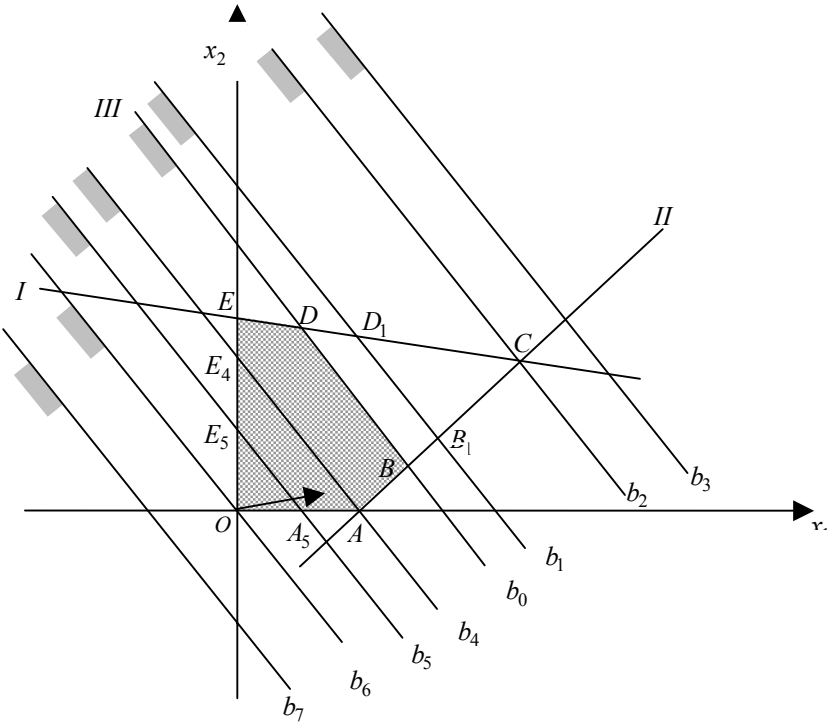


Figure 6.2

On the other hand, starting with the original right-hand side b_0 and decreasing it to b_4, the polytope changes to OAE_4 with an optimal solution at the degenerate point A and the basis x_1, S_1 and exactly one of the variable x_2, S_2 and S_3. Further reduction of the right–hand side value to b_5 results in the polytope OA_5B_5 with the optimal solution A_5 and basis x_1, S_1, S_2 (note that the constraints *I* and *II* have now become redundant) while reducing b_0 further to b_6 has the feasible set degenerate to the origin O, which, if course, is then also optimal. The basis at this point includes S_1, S_2 and exactly one of the variables x_1, x_2, S_3. Further reducing the value of b_0 to b_7 generates a feasible set that is empty.

The above discussion suggests that in case of changes of a right-hand side value, two cases have to be considered.

Case 1: If the *i*-th slack (or excess) variable S_i (or E_i) is not binding at the present optimal solution (i.e., the variable is in the optimal basis with a nonzero value), then an increase (or decrease) of b_i by some small increment ε will result in he change of the value of the slack (excess) variable, while the values of the other variables will remain unchanged.

Case 2: If the *i*-th slack (or excess) variable is binding at the present optimal solution, then, assuming nondegeneracy, any change of b_i, regardless how small, will result in a change of the solution. The basis, however, will not change within a certain range. Since the polytope is modified by such a change, the existence of feasible solutions is no longer guaranteed.

Consider now the change of the *j*-th unit cost or profit coefficient c_j. Variations in the value of c_j will be illustrated by the diagram in Figure 6.3.

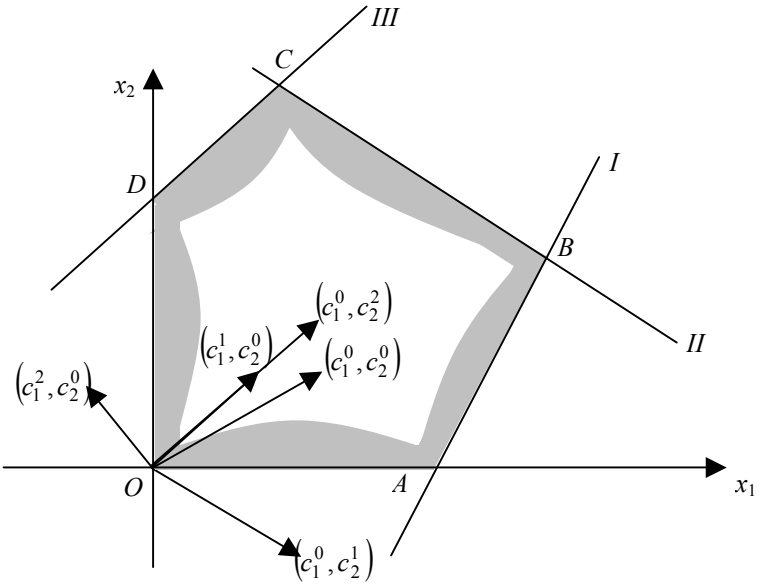


Figure 6.3

The feasible set is shown in Figure 6.3 as the shaded area with the extreme points *O*, *A*, *B*, *C* and *D*, given by the constraints *I*, *II*, *III* in addition to the nonnegativity constraints. Observe that any change in an objective coefficient, regardless how large, does not change the feasible set.

Starting with the objective function Max $z = c_1x_1 + c_2x_2$, the initial the direction of the objective function is $(c_1, c_2) = (c_1^0, c_2^0)$, with $c_1^0, c_2^0 > 0$. The unique optimal solution point is *B*. Keeping c_2 constant at c_2^0 and gradually decreasing the value of c_1, the gradient of the objective function moves in a counterclockwise direction. At a gradient of (c_1^1, c_2^0), the gradient is perpendicular to constraint *II*, so that all

points on the line segment between the extreme points B and C are optimal. Any further change will result in the optimal solution jumping to point C as the unique optimal solution. Once the coefficient c_1 reaches c_1^2, all points on the line segment between the extreme points C and D are optimal. If the coefficient c_1 decreases further, the solution D will remain the unique optimal solution for all finite values of c_1.

Suppose now that we keep the value of c_1 constant at a value of c_1^0 and reduce the value of c_2. For smaller changes, point B remains the unique optimal point until c_2 reaches a value of c_2^1, at which point A and B are alternative optima. Any further reduction of c_2 makes A the unique optimum. Starting from the original c_2^0 and increasing c_2, B remains the unique optimum until c_2 reaches c_2^2, where B and C are both optimal. Any further increase of c_2 makes C the unique optimum.

Note that by changing only c_1, A and O can never become optimal, while if we change only c_2, the points D and O an never be optimal.

Assume now that the objective function of the problem is of the maximization type. We also assume that the model has been optimized and the solution is displayed in the space of variables that are nonbasic at optimum. Note that at this point, the nonnegativity constraints restrict the feasible set to a subset of the nonnegative orthant, while the gradient of the objective functions will always point into the nonpositive orthant. We now distinguish between two cases.

Case 1: If the objective function coefficient of a nonbasic variable decreases, the optimal solution will not change. To see that this result is intuitively obvious, consider a production problem, in which the variables denote the quantities of goods that we manufacture and sell. The decrease of a coefficient c_j then indicates that the profitability of the j-th item has decreased. Since the variable x_j is, by assumption, nonbasic and thus assumes a value of zero, the item was not profitable to begin with and a further decrease in its unit profit will not result in any changes.

If the objective function coefficient of a nonbasic variable x_j increases, there is a certain point where dual degeneracy occurs and thus an alternative optimal production plan including x_j exists. Any further increase in c_j requires x_j to be in the basis and hence increases its production level from the original zero value.

Case 2: If the objective function coefficient of a basic variable increases by a small amount, the basis and solution remain unchanged, while the value of the objective function will increase. If the unit profit c_j increases by more units than specified in the range, the basis will change and, in the absence of primal degeneracy, the solution will also change. More specifically, the value of the basis variable x_j will never decrease; if no primal degeneracy is present in the current

optimal solution, its value will increase. Again, this result is intuitively clear if we consider our production problem: if the price increase of a basic variable, i.e., a variable that is sufficiently profitable to be included in the solution, the product becomes even more profitable and its production level will never decrease. In fact, it will usually increase at the expense of other, less profitable, items.

A sufficiently small decrease of an objective function coefficient of a basic variable x_j will not change the basis or the solution, but reduce the value of the objective function. However, if the change is substantial enough, then the variable x_j will leave the basis, so that the value of the variable decreases to zero.

Having discussed the effects of changing right-hand side values and objective coefficients graphically, the next two sections will examine the algebraic calculations required to obtain exact numerical results for the case of changing parameters. In general, sensitivity analyses are performed in two steps after the optimal tableau has been obtained. In Step 1, we update the parameters with respect to the given changes, and in Step 2, we determine the range of those changes for which the current optimal tableau remains feasible and/or optimal. If a parameter is changed to a value outside its range, we may use parametric programming to determine the new optimal solution for some or all values of c_j, b_i, or a_{ij} between $-\infty$ and $+\infty$. This will be Step 3.

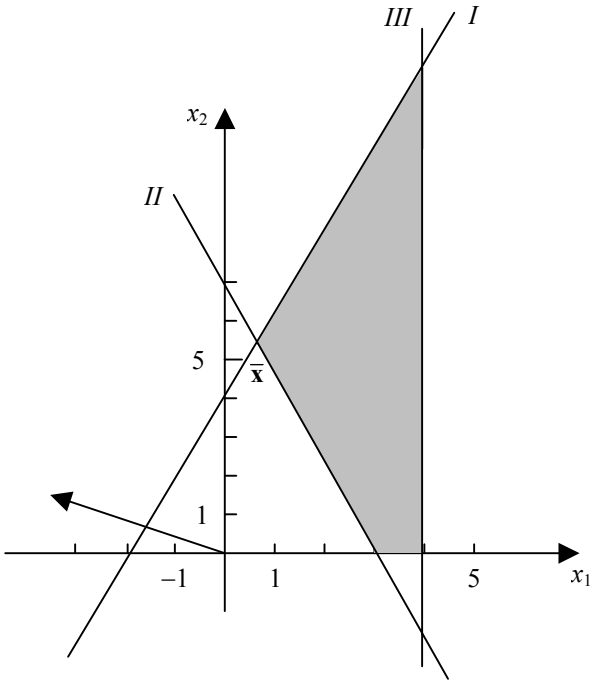


Figure 6.4

Throughout the next sections, we will analyze the following linear programming problem:

$$
\begin{array}{lrll}
\text{P: Max } z = & -3x_1 + 1x_2 & & \\
\text{s.t.} & -2x_1 + 1x_2 \leq & 4 & (I) \\
& 7x_1 + 3x_2 \geq & 21 & (II) \\
& x_1 \leq & 4 & (III) \\
& x_1, \quad x_2 \geq & 0. &
\end{array}
$$

A graphical representation of the problem is shown in Figure 6.4.

Assigning slack, excess, and artificial variables to the left-hand sides of the constraints, so that their subscripts indicate the number of the constraint they are associated with, the initial and final tableaus of the problem are then as follows.

$T^{initial}$:

	x_1	x_2	S_1	E_2	A_2	S_3	1
	–2	1	1	0	0	0	4
	7	3	0	–1	1	0	21
	1	0	0	0	0	1	4
gof	3	–1	0	0	0	0	0
aof	–7	–3	0	1	0	0	–21

$T^{optimal}$:

	x_1	x_2	S_1	E_2	S_3	1
	0	1	$\frac{7}{13}$	$-\frac{2}{13}$	0	$5\frac{5}{13}$
	1	0	$-\frac{3}{13}$	$-\frac{1}{13}$	0	$\frac{9}{13}$
	0	0	$\frac{3}{13}$	$\frac{1}{13}$	1	$3\frac{4}{13}$
gof	0	0	$\frac{16}{13}$	$\frac{1}{13}$	0	$3\frac{4}{13}$

6.2 Changes of the Right-Hand Side Values

This section illustrates how we can obtain information regarding the effects of changes of the right-hand side values on the optimal solution, with no more information than the original problem formulation and optimal tableau. For most of this section, we assume that only a single right-hand side, the i-th, changes. As usual, let the original right-hand values be $[b_1,\ldots,b_m]^T$ and denote by Δb_i the change of the i-th right-hand side, where Δb_i can be positive or negative.

If the change of the i-th right hand side value has been anticipated, we could include an unspecified change Δb_i already in the initial tableau and carry this change through the simplex iterations in the same way as all other elements are; they are, however, not considered (i.e., assumed to be zero) for the selection of

pivot row and/or column. The final tableau is then optimal for the unperturbed problem where $\Delta b_i = 0$. At his point it is then possible to determine the ranges for the change. While such a procedure is certainly legitimate, it is also unnecessary, as all the important information can be gleaned from the final tableau, even if the changes were not anticipated.

If the anticipated change of the i-th right-hand side value Δb_i were to be included in the initial tableau, the right-hand side would be as shown in Table 6.1

Table 6.1

1
b_1
b_2
.
.
b_{i-1}
$b_i + \Delta b_i$
b_{i+1}
.
.
.
b_m

or, writing it as two right-hand sides,

1	Δb_i
b_1	0
b_2	0
.	.
.	.
b_{i-1}	0
b_i	1
b_{i+1}	0
.	.
.	.
.	.
b_m	0

Notice that the right-hand side of the changes, i.e., the column under Δb_i, is a unit vector. In particular, it is the same unit vector we can find in the slack, artificial, or excess variable that was added to the i-th constraint (in case of an excess variable, it is the negative of that vector). For simplicity, assume that the right-hand side of the changes equals that of a slack variable. Denote the column under that slack variable by $\mathbf{a}_{\bullet n+i}$. In general, if two columns in a tableau are identical in any one tableau, then they will stay identical after any number of iterations. This means that the second right-hand side column for Δb_i does not have to be added to the tableau explicitly since its entries are identical to the ones in column $\mathbf{a}_{\bullet n+i}$. Note that in case of the i-th constraint being an equation, the artificial variable A_i must be kept in the tableau, otherwise sensitivity analyses on b_i cannot be performed without recalculating the appropriate information.

In order to formalize our discussion, we will define the following parameters:

b_i denotes the i-th component of the original right-hand side,
b_i^* is the i-th component of the right–hand side value in the optimal tableau,
$\mathbf{a}_{\bullet n+i} = e_i$ is the column under the i-th slack or artificial variable in the original tableau, and

$\mathbf{a}^*_{\bullet n+i}$ denotes the column in the optimal tableau under the i-th slack or artificial variable (or the column under the excess variable with the signs exchanged).

$\overline{\Delta} b_i$ is the largest value of Δb_i, for which $b^*_k + a^*_{k,n+i}\overline{\Delta} b_i \geq 0 \ \forall \ k$=1, …, m, and

$\underline{\Delta} b_i$ is the smallest value of Δb_i for which $b^*_k + a^*_{k,n+i}\underline{\Delta} b_i \geq 0 \ \forall \ k = 1, \ldots, m$.

Sensitivity Analysis: Right-Hand Side Values

Step 1: (Updating of the right-hand side column): The updated right-hand side is $b^*(\Delta b_i) = b^* + \mathbf{a}^*_{\bullet n+i}\,\Delta b_i$.

Step 2: Determine the range $[\underline{\Delta} b_i; \overline{\Delta} b_i]$ for the changes by using the set of inequalities $b^*_k + a^*_{k,n+i}\Delta b_i \geq 0$, $k = 1,..., m$, as well as the range $[\underline{b_i}; \overline{b}_i]$, such that $\underline{b_i} = b^*_i + \underline{\Delta} b_i$ and $\overline{b_i} = b^*_i + \overline{\Delta} b_i$.

If any specific change $\hat{\Delta} b_i$ is given, we proceed as follows. If $\hat{\Delta} b_i \in [\underline{\Delta} b_i; \overline{\Delta} b_i]$, then the optimal basis remains optimal; set $\Delta b_i := \hat{\Delta} b_i$ in the updated right-hand side and the new optimal solution has been obtained. Otherwise, the former optimal basis is no longer optimal. Setting $\Delta b_i := \hat{\Delta} b_i$ in the updated right-hand side results in at least one negative value. A step with the dual simplex method see Section 5.1) must be carried out to reoptimize the problem.

Example 1: Consider the problem introduced at the end of the previous section and perform a sensitivity analysis on the first right-hand side. The slack variable S_1 was added to the first constraint, so that $\mathbf{a}^*_{\bullet n+1} = [1, 0, 0]^T$ and the updated right-hand side vector is

$$b^*(\Delta b_1) = \begin{bmatrix} 70/13 + 7/13\,\Delta b_1 \\ 9/13 - 3/13\,\Delta b_1 \\ 43/13 + 3/13\ \Delta b_1 \\ - - - - - \\ 43/13 + 16/13\ \Delta b_1 \end{bmatrix}.$$

Feasibility now requires that all right-hand side values remain nonnegative, so that the requirements are $70/13 + 7/13\ \Delta b_1 \geq 0$, $9/13 - 3/13\Delta b_1 \geq 0$, and $43/13 + 3/13\Delta b_1 \geq 0$, or $\Delta b_1 \geq -10$, $\Delta b_1 \leq 3$, and $\Delta b_1 \geq -14\frac{1}{3}$, so that $\Delta b_1 \in [-10, 3]$ and thus $b_1 \in [-6, 7]$. In other words, as long as the first right-hand side value assumes

a value between –6 and 7, the basis shown in the optimal tableau, i.e., the basis that consists of x_1, x_2, and S_3, remains optimal. If it has been determined that the first right-hand side value increases by, say, 2 units, then $\Delta\hat{b}_1 = 2$, and the updated right-hand side is

$$b^*(\Delta b_1) = b^*(2) = \begin{bmatrix} 70/13 + 7/13(2) \\ 9/13 - 3/13(2) \\ 43/13 + 3/13(2) \\ - - - \\ 43/13 + 16/13(2) \end{bmatrix} = \begin{bmatrix} 6\frac{6}{13} \\ \frac{3}{13} \\ 3\frac{10}{13} \\ - \\ 5\frac{10}{13} \end{bmatrix},$$

so that $\bar{x}_1 = \frac{3}{13}, \bar{x}_2 = 6\frac{6}{13}, \bar{S}_3 = 3\frac{10}{13}$, and all other variables equal zero. The new value of the objective function is $\bar{z} = 5\frac{10}{13}$.

Suppose now that the change of the first right-hand side value is $\Delta b_1 > 3$. In that case, the second right-hand side value becomes negative and we have to perform an iteration with the dual simplex method. The pivot in the second row is in the E_2 column and after one iteration we obtain tableau T^1,

T^1:

x_1	x_2	S_1	E_2	S_3	1
–2	1	1	0	0	4
–13	0	3	1	0	–9
1	0	0	0	1	4
1	0	1	0	0	4

Updating the right-hand side again with the S_1 column we obtain

$$b^*(\Delta b_1) = \begin{bmatrix} 4 + 1\Delta b_1 \\ -9 + 3\Delta b_1 \\ 4 \\ - - - - \\ 4 + \Delta b_1 \end{bmatrix}.$$

Requiring nonnegativity of all right-hand sides (but not the value of the objective function, of course) results in $4 + \Delta b_1 \geq 0$ and $-9+3\Delta b_1 \geq 0$ or $\Delta b_1 \geq -4$ and $\Delta b_1 \geq 3$, so that tableau T^1 is optimal as long as $\Delta b_1 \in [3, \infty[$, or, equivalently, as long as $b_1 \in [7, \infty[$. Again, the optimal solution for any specific change of the first right-hand side value within this range can easily be calculated by setting Δb_1 to the appropriate value.

Back now to the optimal tableau $T^{optimal}$ and the associated range $\Delta b_1 \in [-10, 3]$. Suppose now that the change is $\Delta b_1 < -10$. In that case, the first right-hand side value in $T^{optimal}$ will be negative and again, a dual simplex step is required. Performing the step leads to the tableau

T^2:

x_1	x_2	S_1	E_2	S_3	1
0	$-\frac{13}{2}$	$-\frac{7}{2}$	1	0	–35
1	$-\frac{1}{2}$	$-\frac{1}{2}$	0	0	–2
0	$\frac{1}{2}$	$\frac{1}{2}$	0	1	6
0	$\frac{1}{2}$	$\frac{3}{2}$	0	0	6

Again, the S_1 column is used to update the right-hand side value, so that we obtain

$$b^*(\Delta b_1) = \begin{bmatrix} -35 - \frac{7}{2}\Delta b_1 \\ -2 - \frac{1}{2}\Delta b_1 \\ 6 + \frac{1}{2}\Delta b_1 \\ ---- \\ 6 + \frac{3}{2}\Delta b_1 \end{bmatrix}.$$

Applying the nonnegativity constraints to the right-hand sides results in $-35 - \frac{7}{2}\Delta b_1 \geq 0$, $-2 - \frac{1}{2}\Delta b_1 \geq 0$, and $6 + \frac{1}{2}\Delta b_1 \geq 0$, resulting in $\Delta b_1 \leq -10$, $\Delta b_1 \leq -4$, and $\Delta b_1 \geq -12$, so that T^2 is optimal within the range $\Delta b_1 \in [-12, -10]$, or, equivalently, as long as $b_1 \in [-8, -6]$. For $b_1 < -8$, the problem has no feasible solution.

We can now summarize our findings in Table 6.2, where the original value of $b_1 = 4$ corresponds to $\Delta b_1 = 0$.

The information in Table 6.2 can be used to plot the so-called *perturbation function* as shown in Figure 6.5, where the optimal value of the objective function is plotted as a function of the right-hand side value b_1.

The slopes of the perturbation function in Figure 6.5 in the three intervals, in which feasible solutions exists, are $1\frac{1}{2}$, $1\frac{3}{13}$, and 1, respectively (from left to right). It can be shown in general that perturbation functions $\bar{z}(b_1)$ are piecewise linear and concave for maximization problems and piecewise linear and convex for minimization problems.

Table 6.2

Range of Δb_1	Range of b_1	Optimal solution (basic variables only)	Value of the objective function
$\Delta b_1 \in \,]-\infty, -12]$	$]-\infty, -8]$	There exists no feasible solution	
$\Delta b_1 \in [-12, -10]$	$[-8, -6]$	$\bar{x}_1 = -2 - \frac{1}{2}\Delta b_1$ $\bar{E}_2 = -35 + \frac{7}{2}\Delta b_1$ $\bar{S}_3 = 6 + \frac{1}{2}\Delta b_1$	$\bar{z} = 6 + \frac{3}{2}\Delta b_1$
$\Delta b_1 \in [-10, 3]$	$[-6, 7]$	$\bar{x}_1 = \frac{9}{13} - \frac{3}{13}\Delta b_1$ $\bar{x}_2 = \frac{70}{13} + \frac{7}{13}\Delta b_1$ $\bar{S}_3 = \frac{43}{13} + \frac{3}{13}\Delta b_1$	$\bar{z} = 3\frac{4}{13} + \frac{16}{13}\Delta b_1$
$\Delta b_1 \in [3, \infty\,[$	$[7, \infty\,[$	$\bar{x}_2 = 4 + \Delta b_1$ $\bar{E}_2 = -9 + 3\Delta b_1$ $\bar{S}_3 = 4$	$\bar{z} = 4 + \Delta b_1$

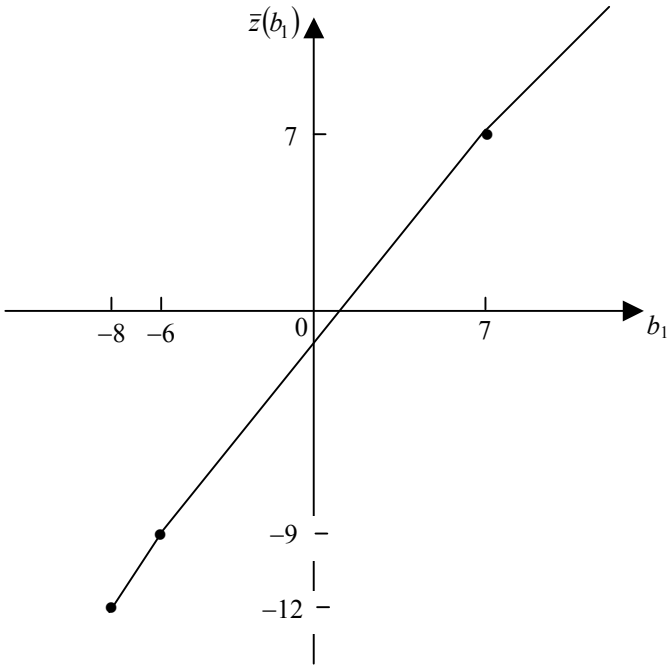


Figure 6.5

Also notice that the shadow price of the first constraint (i.e., the indicator under the S_1 column) equals the slope of the perturbation function in each of the intervals. In other words, as long as the change is strictly within any of the intervals, there is a unique shadow price associated with each resource. Consider now any of the boundaries between two intervals. At any such point, the change off the right-hand side value is such that it results in primal degeneracy, so that there are two different bases with the same numerical solution that belong to the solution. At that point, there are a *right and a left shadow price* for the resource under consideration. In the above example, for $\Delta b_1 = -10$, we have the two optimal bases (x_1, E_2, S_3) and (x_1, x_2, S_3), both with $\bar{x}_1 = 3$ and $\bar{x}_2 = 0$ with $\bar{z} = -9$. The left shadow price of the first resource at that point is 1½, while the right shadow price equals $1\frac{3}{13}$.

Example 2: A similar analysis can be performed when considering changes in the second right-hand side value b_2. Using the E_2-column (as the artificial variable A_2 is basic in the second row in the initial tableau and the E_2 column is identical to it, except for the signs), the updated right-hand side column is now

$$b^*(\Delta b_2) = \begin{bmatrix} 70/13 + 2/13\,\Delta b_2 \\ 9/13 + 1/13\,\Delta b_2 \\ 43/13 - 1/13\ \Delta b_2 \\ - - - - - \\ 43/13 - 1/13\ \Delta b_2 \end{bmatrix},$$

from which we find that the feasibility conditions are $\Delta b_2 \geq -35$, $\Delta b_2 \geq -9$, and $\Delta b_2 \leq 43$, resulting in $\Delta b_2 \in [-9, 43]$ and $b_2 \in [12, 64]$. For a specific change of b_2, e.g., a decrease of the second right-hand side value from the original 21 to, say, 15 (i.e., $\Delta b_2 = -6$), the original optimal basis remains optimal. The updated right-hand side vector is then

$$b^*(\Delta b_2) = b^*(-6) \begin{bmatrix} 70/13 + 2/13(-6) \\ 9/13 + 1/13(-6) \\ 43/13 - 1/13\ (-6) \\ - - - - - \\ 43/13 - 1/13\ (-6) \end{bmatrix} = \begin{bmatrix} 4\frac{6}{13} \\ \frac{3}{13} \\ 3\frac{10}{13} \\ - \\ 3\frac{10}{13} \end{bmatrix},$$

so that the new optimal solution is $\bar{x}_1(b_2 = 15) = \frac{3}{13}$, $\bar{x}_2(b_2 = 15) = 4\frac{6}{13}$, $\bar{S}_3(b_2 = 15) = 2\frac{10}{13}$ and all other variables equal zero, so that the new value of the objective function is $\bar{z}(b_2 = 15) = 3\frac{10}{13}$.

Using parametric programming, it is again possible to establish optimal solutions for all values of $b_2 \in]-\infty, \infty[$. They are shown here in Table 6.3, where $\Delta b_2 = 0$ corresponds to the original value $b_2 = 21$.

Table 6.3

Range of Δb_2	Range of b_2	Optimal solution	z-value
$\Delta b_2 \in]-\infty, -9]$	$]-\infty, 12]$	$\bar{x}_2 = 4$ $\bar{E}_2 = -9 - \Delta b_2$ $\bar{S}_3 = 4$	$\bar{z} = 4$
$\Delta b_2 \in [-9, 43]$	$[12, 64]$	$\bar{x}_1 = \frac{9}{13} + \frac{1}{13}\Delta b_2$ $\bar{x}_2 = 5\frac{5}{13} + \frac{2}{13}\Delta b_2$ $\bar{S}_3 = 3\frac{4}{13} - \frac{1}{13}\Delta b_2$	$\bar{z} = 3\frac{4}{13} - \frac{1}{13}\Delta b_2$
$\Delta b_2 \in [43, \infty[$	$[51, \infty[$	There exists no feasible solution	—

Example 3: Perform a sensitivity analysis on the third right-hand side value b_3. The slack variable S_3 was originally added to the third constraint and it is used to evaluate the changes of the right-hand side column. The updated right-hand side column is

$$b^*(\Delta b_3) = \begin{bmatrix} 70/13 \\ 9/13 \\ 43/13 + \Delta b_3 \\ ----- \\ 3\frac{4}{13} \end{bmatrix},$$

so that feasibility is guaranteed as long as $43/13 + \Delta b_3 \geq 0$ or, equivalently, $\Delta b_3 \geq -43/13$, so that $\Delta b_3 \in [-43/13, \infty[$ and $b_3 \in [9/13, \infty[$.

Another possibility to perform sensitivity analyses was described by Bradley *et al.* (1977). In contrast to the standard sensitivity analyses performed above, it allows *simultaneous* changes of the right-hand side values (or, similarly, of the objective function coefficient, see the succeeding section). In order to initialize their *100 percent rule*, we have to calculate the intervals of the allowable changes first, i.e., we have to compute $[\underline{\Delta b}_i, \overline{\Delta b}_i] \ \forall \ i$. This information is typically provided by the computer program whenever we choose the "Sensitivity Analysis" option. Let then B^+ denote the set of right-hand sides that are to increase and B^- as the set of right-hand sides that are to decrease. Formally, we define $B^+ = \{i: \hat{\Delta} b_i > 0\}$ and

$B^- = \{i\colon\ \hat{\Delta}b_i < 0\}$. The proportion of the actual change to the allowable change in that direction is then $\frac{\hat{\Delta}b_i}{\overline{\Delta}b_i}$ for $i \in B^+$ and $\frac{\hat{\Delta}b_i}{\underline{\Delta}b_i}$ for $i \in B^-$. The requirement can then be stated as follows:

Lemma 6.1: The optimal basis of a linear program will remain optimal as long as the sum of percentage changes does not exceed 100%, i.e.,

$$\sum_{i\in B^+}\frac{\hat{\Delta}b_i}{\overline{\Delta}b_i} + \sum_{i\in B^-}\frac{\hat{\Delta}b_i}{\underline{\Delta}b_i} \le 1\,.$$

Example 4: Consider again the example above and suppose that the first right-hand side may increase from the original 4 to 6, while the second right-hand side may assume a value of 18 rather than the original 21. This means that $\hat{\Delta}b_1 = 2$ and $\hat{\Delta}b_2 = -3$, so that $1 \in B^+$, while $2 \in B^-$. We then obtain $\frac{2}{3} + \frac{-3}{-9} = 1 = 100\%$, so that the conditions is satisfied and the optimal basis (but not the optional solution) remains unchanged. If instead the first two right-hand sides were to change to 5 and 17, respectively, we obtain $\hat{\Delta}b_1 = 1$ and $\hat{\Delta}b_2 = -4$, resulting in a sum of percentage changes of $\frac{1}{3} + \frac{-4}{-9} = \frac{7}{9} < 1$, so that again, the optimal basis remains unchanged. If, however, the first two right-hand side values were to change to 5½ and 16, respectively, we obtain $\hat{\Delta}b_1 = 1½$ and $\hat{\Delta}b_2 = -5$, so that the sum of percentage changes is $\frac{1½}{3} + \frac{-5}{-9} = \frac{19}{18} > 1$, so that the optimal basis will no longer be optimal and the problem has to be reoptimized with the new right-hand side values.

6.3 Changes of the Objective Function Coefficients

The question posed in this section and the procedure leading to its answer is similar to that discussed in the previous section. Here, we examine the effects of changes of the coefficients in the objective function on the optimal solution. For simplicity, assume that we have a problem in which we are to maximize the objective function $\sum_{k=1}^{n} c_k x_k$. If we were to anticipate changes in one of the objective function coefficients, say, the *j*-th, we could again use the slightly modified objective Max $\sum_{k=1}^{j-1} c_k x_k + (c_j + \Delta c_j)x_j + \sum_{k=j+1}^{n} c_k x_k$, i.e., introducing a slightly perturbed cost coefficient for the *j*-th variable.

Again, it is not necessary to introduce Δc_j in the initial tableau and carry it through all the iterations. Instead, we can recreate the updated objective function from the information obtained in the final tableau without adding anything to it. In order to formalize matters along the lines of the previous section, we introduce the following notation:

c_j denotes the coefficient of the variable x_j, and
c_j^* is the value under x_j in the final tableau (i.e., the opportunity cost of x_j)

We can then update the objective function of the optimal tableau by the following simple steps. First, we add the missing term Δc_j to the appropriate opportunity cost in the final tableau. (Note that adding Δc_j in a tableau with a maximization function means adding $-\Delta c_j$ to the opportunity cost, while in a tableau with a minimization objective, we will add Δc_j to the expression). We now have to distinguish between two cases.

Case 1: x_j is a nonbasic variable. In this case $c_j^*(\Delta c_j) = c_j^* - \Delta c_j \geq 0$ (required for dual feasibility) and an interval for $\Delta c_j \in]-\infty; c_j^*]$ can be determined which completes the sensitivity analysis.

Case 2: x_j is a basic variable that is in the basis in, say, the *i*–th row which currently includes the elements $a_{i\bullet}^*$. Adding Δc_j to the *j*-th component in the objective function destroys the formal definition of a basic variable, which requires that $c_k^* = 0$ for all basic variables x_k. In order to restore a complete basis, we can pivot on the element $a_{ij}^* = 1$. Note that the only changes from this tableau to the next occur in the row for the objective function; in particular we obtain $c^*(\Delta c_j) = c^* + a_{i\bullet}^* \Delta c_j$. Again $c^*(\Delta c_j)$ is the updated objective function and requiring $c_k^*(\Delta c_j) \geq 0 \; \forall \; k$ results in an interval $\Delta c_j \in [\underline{\Delta c}_j; \overline{\Delta c}_j]$, which completes the sensitivity analysis for this case.

The above discussion can be summarized in the following procedure:

Sensitivity Analysis: Objective Function Coefficients (Maximization function)

Step 1: Is x_j a basic variable?
If yes: Go to Step 2.
If no: Subtract Δc_j from c_j^* and determine the interval $\Delta c_j \in]-\infty; c_j^*]$.

Step 2: Determine the updated objective function $c^*(\Delta c_j) = c^* + a^*_{i\bullet}\Delta c_j$, where x_j is in the basis in row i. Compute the interval $\Delta c_j \in [\underline{\Delta c}_j; \overline{\Delta c}_j]$ by using the set of inequalities $c^*_k(\Delta c_j) \geq 0 \; \forall \; k$ as well as the interval $[\underline{c}_j; \overline{c}_j]$, where $\underline{c}_j = c^*_j + \underline{\Delta c}_j$ and $\overline{c}_j = c^*_j + \overline{\Delta c}_j$.

This procedure will be illustrated by means of the example introduced at the beginning of the previous section. For convenience, the optimal tableau T^{optimal} is shown again below.

T^{optimal}:

x_1	x_2	S_1	E_2	S_3	1
0	1	$\frac{7}{13}$	$-\frac{2}{13}$	0	$5\frac{5}{13}$
1	0	$-\frac{3}{13}$	$-\frac{1}{13}$	0	9/13
0	0	$\frac{3}{13}$	$\frac{1}{13}$	1	$3\frac{4}{13}$
0	0	$\frac{16}{13}$	$\frac{1}{13}$	0	$3\frac{4}{13}$

Example 5: Given the same example as before, perform a sensitivity analysis on the variable x_1. In order to do so, we first have to update the objective function by subtracting Δc_1 in the c_1 column in the tableau, followed by a simplex iteration with the pivot element $a^*_{21} = 1$. The result is the objective function row

x_1	x_2	S_1	E_2	S_3	1
0	0	$\frac{16}{13} - \frac{3}{13}\Delta c_1$	$\frac{1}{13} - \frac{1}{13}\Delta c_1$	0	$3\frac{4}{13} + \frac{9}{13}\Delta c_1$

The optimality conditions are then $\frac{16}{13} - \frac{3}{13}\Delta c_1 \geq 0$ and $\frac{1}{13} - \frac{1}{13}\Delta c_1 \geq 0$, or simply $\Delta c_1 \leq 5\frac{1}{3}$ and $\Delta c_1 \leq 1$, so that the solution in T^{optimal}, i.e., $\overline{x}_1 = \frac{9}{13}, \overline{x}_2 = 5\frac{5}{13}$ and $\overline{S}_3 = 3\frac{4}{13}$ with a value of the objective function of $\overline{z} = 3\frac{4}{13}$ remains optimal as long as $\Delta c_1 \in \;]-\infty, 1]$ and hence $c_1 \in \;]-\infty, -2]$. If $\Delta c_1 > 1$, then the indicator in the objective function in the E_2 column becomes negative and the variable E_2 has to be introduced into the basis. The only pivot-eligible element is in the third row, so that the slack variable S_3 leaves the basis. After one pivoting step we obtain the tableau T^3:

T^3:

x_1	x_2	S_1	E_2	S_3	1
0	1	1	0	2	12
1	0	0	0	1	4
0	0	3	1	13	43
0	0	1	0	$-1+\Delta c_1$	$4\Delta c_1$

The optimal solution is now $\bar{x}_1 = 4, \bar{x}_2 = 12$, so that $\bar{z} = 4\Delta c_1$. This tableau is optimal as long as $-1+\Delta c_1 \geq 0$, i.e., T^3 and its solution are optimal for $\Delta c_1 \in [1, \infty[$ or, equivalently, $c_1 \in [-2, \infty[$. This establishes optimal solutions for the entire range of Δc_1.

Example 6: Performing a sensitivity analysis for c_2, we update the objective function by using the first row of the tableau, as x_2 is in the basis in row 1. The updated objective function row is then

x_1	x_2	S_1	E_2	S_3	1
0	0	$\frac{16}{13}+\frac{7}{13}\Delta c_2$	$\frac{1}{13}-\frac{2}{13}\Delta c_2$	0	$3\frac{4}{13}+5\frac{5}{13}\Delta c_2$

The optimality conditions are then $\frac{16}{13}+\frac{7}{13}\Delta c_2 \geq 0$ and $\frac{1}{13}-\frac{2}{13}\Delta c_2 \geq 0$, which lead to $\Delta c_2 \geq -16/7$ and $\Delta c_2 \leq ½$, i.e., $\Delta c_2 \in [-2\frac{2}{7}, ½]$, so that and $c_2 \in [-1\frac{2}{7}, 1½]$. As long as these conditions are satisfied, $T^{optimal}$ and the solution therein remains optimal.

Suppose now that $\Delta c_2 < -\frac{16}{7}$. In this case, the shadow price of S_1 is negative and S_1 must be introduced into the basis in order to restore optimality. The unique pivot in the S_1 column is in the first row and one simplex iteration leads to tableau T^4.

T^4:

x_1	x_2	S_1	E_2	S_3	1
0	$\frac{13}{7}$	1	$-\frac{2}{7}$	0	10
1	$\frac{3}{7}$	0	$-\frac{1}{7}$	0	3
0	$-\frac{3}{7}$	0	$\frac{1}{7}$	1	1
0	$-\frac{16}{7}-\Delta c_2$	0	$\frac{3}{7}$	0	–9

Tableau T^4 will remain optimal, as long as $-\frac{16}{7}-\Delta c_2 \geq 0$, i.e., for all $\Delta c_2 \leq 2\frac{2}{7}$. In other words, tableau T^4 with the solution $\bar{x}_1 = 3, \bar{x}_2 = 0$, $\bar{S}_1 = 10$, $\bar{E}_2 = 0$ and $\bar{S}_3 = 1$, so that $\bar{z} = -9$ as long as $\Delta c_2 \in]-\infty, -2\frac{2}{7}]$, or, equivalently, $c_2 \in]-\infty, -1\frac{2}{7}]$.

Back now to tableau $T^{optimal}$, which was optimal as long as $\Delta c_2 \in [-2\frac{2}{7}, ½]$. If $\Delta c_2 > ½$, then the shadow price of E_2 is negative and the variable E_2 must be introduced into the basis to restore optimality. The pivot element is in the third row and one simplex iteration leads to tableau T^3, except that the objective function row is now

x_1	x_2	S_1	E_2	S_3	1
0	0	$1+\Delta c_2$	0	$-1+2\Delta c_2$	0

which remains optimal as long as $1 + \Delta c_2 \geq 0$ and $-1 + 2\Delta c_2 \geq 0$, or, equivalently, $\Delta c_2 \geq -1$ and $\Delta c_2 \geq ½$, so that the tableau and its basis remains optimal as long as $\Delta c_2 \in [½, \infty[$.

These results are summarized in Table 6.4, where $\Delta c_2 = 0$ corresponds to the original value $c_2 = 1$.

Table 6.4

Range of Δc_2	Range of c_2	Optimal solution	Value of the objective function
$\Delta c_2 \in$ $]-\infty, -2\frac{2}{7}]$	$]-\infty, -1\frac{2}{7}]$	$\bar{x}_1 = 3,$ $\bar{S}_1 = 10,\ \bar{S}_3 = 1$	$\bar{z} = -9$
$\Delta c_2 \in [-2\frac{2}{7},\ \frac{1}{2}]$	$[-1\frac{2}{7},\ 1\frac{1}{2}]$	$\bar{x}_1 = \frac{9}{13},\ \bar{x}_2 = 5\frac{5}{13}$ $\bar{S}_3 = 3\frac{4}{13}$	$\bar{z} = 3\frac{4}{13} + 5\frac{5}{13}\Delta c_2$
$\Delta c_2 \in [\frac{1}{2}, \infty[$	$[1\frac{1}{2}, \infty[$	$\bar{x}_1 = 4,\ \bar{x}_2 = 12,$ $\bar{E}_3 = 43$	$\bar{z} = 12\Delta c_2$

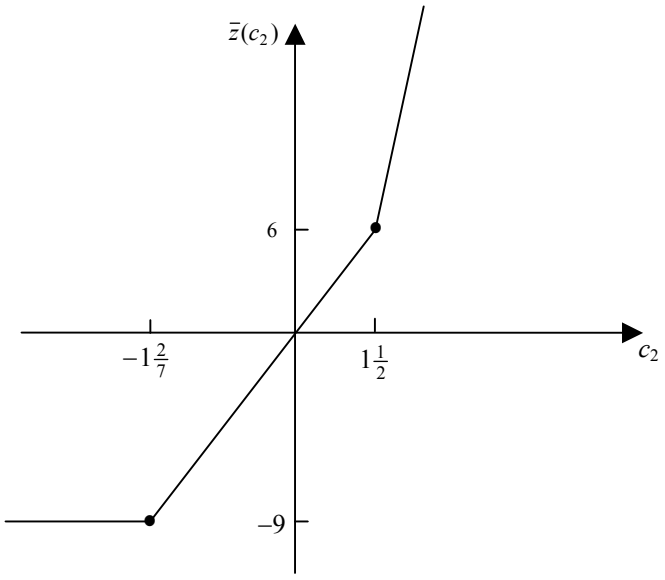


Figure 6.6

Given this information, we could again construct the perturbation graph as shown in Figure 6.6. The perturbation function has slopes of 0, $5\frac{5}{13}$, and 12 in its three linear pieces (from left to right) and it is thus convex.

With the help of Table 6.4 and Figure 6.6, we can now easily determine a solution and its associated value of the objective function for any finite value of Δc_2 or c_2.

For instance, if c_2 were to equal –1, the first row of Table 6.4 applies, $\Delta c_2 = -2$ and the optimal solution is $\bar{x}_1 = 3, \bar{x}_2 = 0$ with $\bar{z} = -9$. If c_2 were to equal 3 (or, equivalently, $\Delta c_2 = 2$), then the third row in Table 6.4 applies and the optimal solution is $\bar{x}_1 = 4, \bar{x}_2 = 12$ with $\bar{z} = 12\Delta c_2 = 24$.

We conclude this section by applying the "100% rule" which is handled similarly as the same rule applied to changes of the right-hand side values discussed in the previous section. Define sets $C^+ = \{j\colon \hat{\Delta}c_j > 0\}$ and $C^- = \{j\colon \hat{\Delta}c_j < 0\}$. The proportion of the actual change to the allowable change in that direction is then $\frac{\hat{\Delta}c_j}{\overline{\Delta c}_j}$ for $j \in C^+$ and $\frac{\hat{\Delta}c_j}{\underline{\Delta c}_j}$ for $j \in C^-$. The requirement is then

Lemma 6.2: The optimal basis of a linear program will remain optimal as long as the sum of percentage changes does not exceed 100%, i.e.,

$$\sum_{j\in C^+} \frac{\hat{\Delta}c_j}{\overline{\Delta c}_j} + \sum_{j\in C^-} \frac{\hat{\Delta}c_j}{\underline{\Delta c}_j} \le 1 .$$

Example: Apply the 100% rule to the example above. Recall that the tableau $T^{optimal}$ remains optimal as long as $\Delta c_1 \in]-\infty, 1]$ and $\Delta c_2 \in [-2\frac{2}{7}, ½]$. Assume now that the expected changes are $\hat{\Delta}c_1 > 0$ and $\hat{\Delta}c_2 < 0$, so that $C^+ = \{1\}$ and $C^- = \{2\}$. Hence, in our example we obtain the feasibility condition $\frac{\hat{\Delta}c_1}{1} + \frac{\hat{\Delta}c_2}{-2\frac{2}{7}} \le 1$.

For instance, if $\hat{\Delta}c_1 = ½$ and $\hat{\Delta}c_2 = -1$. we obtain ½ + $\frac{1}{\frac{16}{7}}$ = 15/16 < 1, so that the change will leave the optimal basis unchanged.

6.4 Sensitivity Analyses in the Presence of Degeneracy

This section will address the issue of performing sensitivity analyses in the presence of primal and dual degeneracy. We will conduct our discussion by means

of a numerical example. Throughout this section, we will be considering the model

$$\begin{aligned} \text{P: Max } z &= 6x_1 + 4x_2 \\ \text{s.t.} \quad & 4x_1 + 6x_2 \le 24 \\ & 5x_1 + 5x_2 \le 24 \\ & 6x_1 + 4x_2 \le 24 \\ & x_1, \quad x_2 \ge 0, \end{aligned}$$

whose optimal simplex tableaus are

T^{opt1}:

x_1	x_2	S_1	S_2	S_3	1
0	$\frac{10}{3}$	1	0	$-\frac{2}{3}$	8
0	$\frac{5}{3}$	0	1	$-\frac{5}{6}$	4
1	$\frac{2}{3}$	0	0	$\frac{1}{6}$	4
0	0	0	0	1	24

and an alternative optimal solution is

T^{opt2}:

x_1	x_2	S_1	S_2	S_3	1
0	0	1	–2	1	0
0	1	0	$\frac{3}{5}$	$-\frac{1}{2}$	$\frac{12}{5}$
1	0	0	$-\frac{2}{5}$	$\frac{1}{2}$	$\frac{12}{5}$
0	0	0	0	1	24

as well as the same solution with a different basis shown in the tableau

T^{opt3}:

x_1	x_2	S_1	S_2	S_3	1
0	1	$\frac{3}{10}$	0	$-\frac{1}{5}$	$\frac{12}{5}$
0	0	$-\frac{1}{2}$	1	$-\frac{1}{2}$	0
1	0	$-\frac{1}{5}$	0	$\frac{3}{10}$	$\frac{12}{5}$
0	0	0	0	1	24

It is apparent that the problem exhibits primal as well as dual degeneracy. We can now perform sensitivity analyses as usual.

Example 7: Perform a sensitivity analysis on the objective function coefficient c_1.

Using the standard updating techniques as discussed in Sections 6.2 and 6.3, we obtain the following ranges:

- The basis x_1, S_1, S_2 with $\bar{x}_1 = 4, \bar{x}_2 = 0$ and $\bar{z} = 24$ is optimal for $\Delta c_1 \in [0, \infty[$ (via T^{opt1}),
- the basis x_1, x_2, S_1 with $\bar{x}_1 = \frac{12}{5}$, $\bar{x}_2 = \frac{12}{5}$ and $\bar{z} = 24$ is optimal for $\Delta c_1 \in [-2, 0]$ (T^{opt2}), and
- the basis x_1, x_2, S_2 with $\bar{x}_1 = \frac{12}{5}$, $\bar{x}_2 = \frac{12}{5}$ and $\bar{z} = 24$ is optimal for $\Delta c_1 \in [-\frac{10}{3}; 0]$ (via T^{opt3}).

Worse, consider the tableau we obtain from T^{opt2} with $\Delta c_2 < -2$. It is shown as T^1.

T^1:

x_1	x_2	S_1	S_2	S_3	1
0	0	1	–2	1	0
0	1	$\frac{1}{2}$	$-\frac{2}{5}$	0	$\frac{12}{5}$
1	0	$-\frac{1}{2}$	$\frac{3}{5}$	0	$\frac{12}{5}$
0	0	–1	2	0	24

which remains optimal for $\Delta c_1 \in [-\frac{10}{3}, -2]$. Finally, the tableau obtained from T^{opt3} with $\Delta c_1 < \frac{10}{3}$ is shown as T^2.

T^2:

x_1	x_2	S_1	S_2	S_3	1
$\frac{10}{3}$	0	$-\frac{2}{3}$	0	1	8
$\frac{2}{3}$	1	$\frac{1}{6}$	0	0	4
$\frac{5}{3}$	0	$-\frac{5}{6}$	1	0	4
$-\frac{10}{3}$	0	$\frac{2}{3}$	0	0	16

The tableau T^2 is optimal for $\Delta c_1 \in [-\infty, -\frac{10}{3}]$. The perturbation function is shown in Figure 6.7 Part of that function, *viz*., the pieces between $-\frac{10}{3}$ and –2 and the piece between –2 and 0, are determined by two tableaus and bases each.

It is interesting to note that, as far as solutions are concerned,

- the solution $\bar{x}_1 = 0$ and $\bar{x}_2 = 4$ with $\bar{z} = 16$ is optimal for $\Delta c_1 \in]-\infty, -\frac{10}{3}]$,
- the solution $\bar{x}_1 = \bar{x}_2 = \frac{12}{5}$ with $\bar{z} = 24 + \frac{12}{5}\Delta c_1$ is optimal for $\Delta c_1 \in [-\frac{10}{3}, 0]$, and

- the solution $\bar{x}_1 = 4$ and $\bar{x}_2 = 0$ with $\bar{z} = 24 + 4\Delta c_1$ is optimal for $\Delta c_1 \in [0, \infty[$.

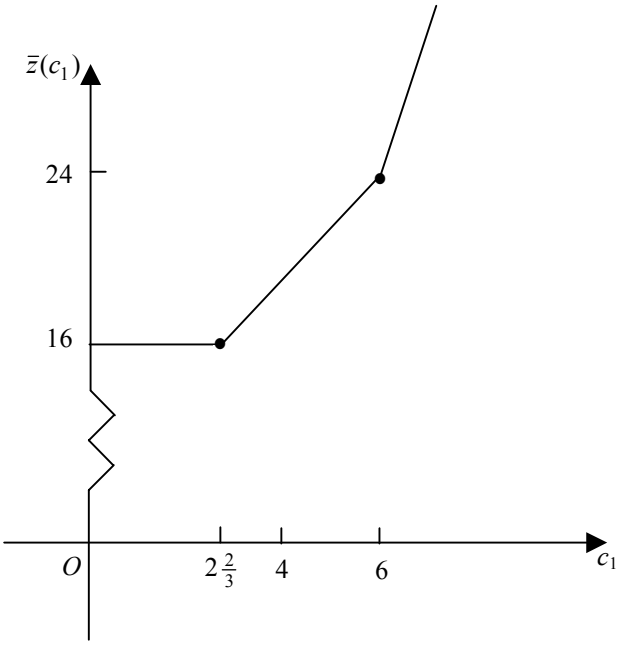


Figure 6.7

Information that stresses optimal solutions rather than optimal bases will be somewhat harder to obtain from the sensitivity analyses provided by computer packages, but it is much more useful to the decision maker, so that the extra effort will be very much worthwhile.

6.5 Addition of a Constraint

This section will describe the addition of a constraint to an optimal tableau. In fact, the same procedure could be applied to add a constraint to any tableau, but almost all applications of this procedure start with an optimal tableau and solution. Most applications are found when a model is built gradually from a simple formulation with few variables and constraints to a highly complex structure with, typically, thousands or even hundreds of thousands of variables. As long as the model is small, analysts will simply re-solve the modified model. However, when the model is very large, it may be worthwhile to add a constraint to the present optimal tableau and continue with a "warm start." Another application occurs in integer programming, when constraints (so-called cutting planes) are gradually added to the problem in the solution process.

Before discussing specifics concerning the addition of a constraint, we would like to offer some general comments regarding the addition or deletion of constraints and variables. If a constraint is deleted, the formulation gets "looser," so that the new optimal solution may be better than the present solution. A similar effect is obtained when adding another variable to the model. A new variable provides the possibility of new activities or courses of action (if this is not desired, we can simply set the new variable to zero and are left with the previous solution), so that the value of the objective function will either remain unchanged or will improve. On the other hand, deleting a variable will eliminate possibilities and may result in a deterioration of the objective function. Similarly, adding a constraint will impose new restrictions on the model that may deteriorate the objective function.

The first step when adding a constraint is to determine whether or not the present (optimal) solution satisfies this new constraint. If so, the solution remains optimal and the process terminates. Suppose now that the present solution does not satisfy the new constraint. The first task is then to add the constraint to the simplex tableau, which will require to express the basic variables in the new constraint in terms of nonbasic variables. This will result in a new row in the simplex tableau that has a negative right-hand side value. Note that this will always be the case, since, by assumption, the present solution violates the new constraint.

Once this is accomplished, we add the new constraint to our optimal tableau and perform dual simplex iterations with the first pivot being chosen in the row that belongs to the new constraint, which is the only pivot-eligible row.

This process will be illustrated by the following

Example: Consider again the linear programming problem at the end of Section 6.1. For convenience, the problem and its final tableau are restated here.

$$
\begin{aligned}
\text{P: Max } z = &-3x_1 + 1x_2 \\
\text{s.t.} \quad &-2x_1 + 1x_2 \leq 4 \\
&7x_1 + 3x_2 \geq 21 \\
&x_1 \leq 4 \\
&x_1, \quad x_2 \geq 0.
\end{aligned}
$$

The optimal tableau of the problem is then

T^{optimal}:

x_1	x_2	S_1	E_2	S_3	1
0	1	$\frac{7}{13}$	$-\frac{2}{13}$	0	$5\frac{5}{13}$
1	0	$-\frac{3}{13}$	$-\frac{1}{13}$	0	$\frac{9}{13}$
0	0	$\frac{3}{13}$	$\frac{1}{13}$	1	$3\frac{4}{13}$
0	0	$\frac{16}{13}$	$\frac{1}{13}$	0	$3\frac{4}{13}$

Suppose now that we are to add the constraint $3x_1 + 2x_2 \le 15$ to the system. A quick check reveals that the present solution does, indeed, satisfy this constraint, so that $T^{optimal}$ remains optimal and no further action is necessary.

Let now the new constraint be $3x_1 + 2x_2 \le 10$. This constraint is violated by the present optimal solution, so that we have to add it to $T^{optimal}$. In order to do so, it is required that we express the basic variables x_1 and x_2 in terms of nonbasic variables, as simply adding the constraint to the tableau would result in nonzero elements under basic variables, thus destroying the unit vectors that must be found in these columns. In this example, the second and first rows of the tableau read

$$x_1 - \tfrac{3}{13}S_1 - \tfrac{1}{13}E_2 + \tfrac{1}{13}A_2 = \tfrac{9}{13} \text{ and } x_2 + \tfrac{7}{13}S_1 - \tfrac{2}{13}E_2 + \tfrac{2}{13}A_2 = \tfrac{70}{13},$$

where the artificial variable has been added for completeness. Solving the two equations for x_1 and x_2, respectively, the new constraint can be written as

$$3(\tfrac{9}{13} + \tfrac{3}{13}S_1 + \tfrac{1}{13}E_2) + 2(\tfrac{70}{13} - \tfrac{7}{13}S_1 + \tfrac{2}{13}E_2) \le 10,$$

which can be written as $-\tfrac{5}{13}S_1 + \tfrac{7}{13}E_2 \le -\tfrac{37}{13}$. Adding a column with the new slack variable S_4 and this new constraint to the tableau $T^{optimal}$ results in

T^1:

x_1	x_2	S_1	E_2	S_3	S_4	1
0	1	$\frac{7}{13}$	$-\frac{2}{13}$	0	0	$5\frac{5}{13}$
1	0	$-\frac{3}{13}$	$-\frac{1}{13}$	0	0	$\frac{9}{13}$
0	0	$\frac{3}{13}$	$\frac{1}{13}$	1	0	$3\frac{4}{13}$
0	0	$-\frac{5}{13}$	$\frac{7}{13}$	0	1	$-\frac{37}{13}$
0	0	$\frac{16}{13}$	$\frac{1}{13}$	0	0	$3\frac{4}{13}$

As the solution is no longer feasible, we perform one dual simplex step with the new row as the pivot row, resulting in tableau T^2.

T^2:

x_1	x_2	S_1	E_2	S_3	S_4	1
0	1	0	$\frac{3}{5}$	0	0	$\frac{7}{5}$
1	0	0	$-\frac{2}{5}$	0	0	$\frac{12}{5}$
0	0	0	$\frac{2}{5}$	1	0	$\frac{8}{5}$
0	0	1	$-\frac{7}{5}$	0	1	$\frac{37}{5}$
0	0	0	$\frac{9}{5}$	0	0	$-\frac{29}{5}$

The new solution is $\bar{x}_1 = 2.4$, $\bar{x}_2 = 1.4$, $\bar{S}_1 = 7.4$, $\bar{E}_2 = 0$, $\bar{S}_3 = 1.6$, and $\bar{S}_4 = 0$, so that $\bar{z}$ = –5.8. Notice the dramatic deterioration of the value of the objective function from about 3.3 to –5.8, even though the left-hand side of the new constraint was 12.85 with the old optimal solution and only a minor decrease to no more than ten was required.

While the above updating procedure was correct, it was nonetheless awkward. We can, however, use simple pivoting to update the optimal tableau. Simply adding the new constraint and the new slack variable S_4 to $T^{optimal}$ results in tableau T^3 below.

T^3:

x_1	x_2	S_1	E_2	S_3	S_4	1
0	1	$\frac{7}{13}$	$-\frac{2}{13}$	0	0	$5\frac{5}{13}$
1	0	$-\frac{3}{13}$	$-\frac{1}{13}$	0	0	$\frac{9}{13}$
0	0	$\frac{3}{13}$	$\frac{1}{13}$	1	0	$3\frac{4}{13}$
3	2	0	0	0	1	10
0	0	$\frac{16}{13}$	$\frac{1}{13}$	0	0	$3\frac{4}{13}$

Notice that the columns of the basic variables x_1 and x_2 no longer are the required unit vectors. In order to restore a proper tableau, we carry out two standard simplex iterations with the pivots being the "1" elements in the basic columns. Note that in each of these simplex iterations, only the coefficients in the new row will change, thus allowing for easier calculations. In our example, the tableau that results after two simplex iterations is the same as T^1.

While the above procedure is certainly valid, the updating of the tableau is not practical. Recall that the system of equations in standard form can be written as

$$\mathbf{B}\mathbf{x}_\mathbf{B} + \mathbf{N}\mathbf{x}_\mathbf{N} = \mathbf{b}$$

with the basic and nonbasic matrices **B** and **N** and the vector of basic and nonbasic variables, respectively. (For details, readers are again referred to the discussion in Chapter 3 of this volume. The present solution is then obtained by premultiplying the system of linear equations by $\mathbf{B}^{-1}$, the inverse of the basis matrix **B**, and solving for $\mathbf{x}_\mathbf{B}$, resulting in

$$\mathbf{x}_\mathbf{B} = \mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_\mathbf{N}. \qquad (1)$$

Suppose now that the constraint to be added is $a_{i\bullet}x = b_i$, which can be written as

$$\mathbf{a}_{i\bullet}^{\mathbf{B}}\mathbf{x}_\mathbf{B} + \mathbf{a}_{i\bullet}^{\mathbf{N}}\mathbf{x}_\mathbf{N} = b_i, \qquad (2)$$

where $\mathbf{a}_{i\bullet}^{\mathbf{B}}$ and $\mathbf{a}_{i\bullet}^{\mathbf{N}}$ are the appropriate parts of the left-hand side of the new constraint. Replacing (1) in (2) results in

$$(-\mathbf{a}_{i\bullet}^{\mathbf{B}}\mathbf{B}^{-1}\mathbf{N} + \mathbf{a}_{i\bullet}^{\mathbf{N}})\mathbf{x}_{\mathbf{N}} = b_i - \mathbf{a}_{i\bullet}^{\mathbf{B}}\mathbf{B}^{-1}\mathbf{b}\,,$$

where the term in brackets on the left-hand side indicates the new coefficients of the nonbasic variables in the present tableau, and the updated right-hand side is also available.

In our example, $\mathbf{x}_{\mathbf{B}} = [x_2, x_1, S_3]^T$, $\mathbf{x}_{\mathbf{N}} = [S_1, E_2]^T$, $\mathbf{B} = \begin{bmatrix} 1 & -2 & 0 \\ 3 & 7 & 0 \\ 0 & 1 & 1 \end{bmatrix}$,

$\mathbf{B}^{-1} = \begin{bmatrix} \frac{7}{13} & \frac{2}{13} & 0 \\ -\frac{3}{13} & \frac{1}{13} & 0 \\ \frac{3}{13} & -\frac{1}{13} & 1 \end{bmatrix}$, $\mathbf{a}_{i\bullet}^{\mathbf{B}} = [2, 3, 0]$, $\mathbf{a}_{i\bullet}^{\mathbf{N}} = [0, 0]$, and $b_i = 10$. The coefficients of the nonbasic variables in the tableau are then

$-\mathbf{a}_{i\bullet}^{\mathbf{B}}\mathbf{B}^{-1}\mathbf{N} + \mathbf{a}_{i\bullet}^{\mathbf{N}} = [-\frac{5}{13}, \frac{7}{13}]$, and the right-hand side value is

$b_i - \mathbf{a}_{i\bullet}^{\mathbf{B}}\mathbf{B}^{-1}\mathbf{b} = -\frac{37}{13}$. This is the desired result. Now the row can be added and simplex iterations can be performed as required.

6.6 Economic Analysis of an Optimal Solution

To put the sensitivity analyses concepts to use, we will formulate a small numerical example and provide a variety of managerial answers, indicating where on the printout (with sensitivity option) can be found.

The model involves the manufacture of perfumes. In particular, there are three types of perfumes that we may blend, *viz*., ODOOR, ODDHEUR, and McSMELL. Each of the perfumes is sold in its own 1 fl.oz. bottle, and all of them are blended from bergamot essence and musk, while oil is used as base. The ingredients are available in certain quantities and at certain (predetermined) costs, while the prices of the three products have been set at levels determined by the marketing department. We assume that at the given prices, customers will purchase any quantity we can manufacture. Detailed numerical information is found in Table 6.5.

Table 6.5

	ODOOR	ODDHEUR	McSMELL	Availability (fl.oz.)	Price per fl.oz.
Bergamot essence	.3	.2	.05	at most 2,400 fl.oz.	$50
Musk	.1	.4	.05	at most 600 fl. oz.	$80
Oil	.6	.4	.9	unlimited quantities	$5
Sales price per 1 fl.oz. bottle	$40	$75	$15		

In addition, it has been determined that we should at least make 200 bottles of the prestigious perfume ODDHEUR in order to be mentioned in the appropriate magazines. Furthermore, we should make a total of at least 4,000 bottles in order to show presence in the market.

In order to formulate the problem, we define ODOOR, ODDHEUR, and MCSMELL as the number of bottles of the respective perfumes that are made *and sold*. Before formulating the model, we have to compute the unit profit contributions of the three perfumes. Consider ODOOR. The sales price is $40 per fl.oz., while the costs are calculated on the basis of its ingredients: 0.3 fl.oz. bergamot essence at $50 per fl.oz. costing $15, .1 fl.oz. of musk at $ 80 per fl.oz. costs $8, and .6 fl.oz. oil for $5 per unit, costing $3 for total unit costs of $26; hence a unit profit of $14. The profit contributions of the other perfumes are calculated similarly.

The model can then be formulated as

$$\text{P: Max } z = 14ODOOR + 31ODDHEUR + 4MCSMELL$$

$$\begin{aligned}
\text{s.t.} \quad & .3ODOOR + .2ODDHEUR + .05MCSMELL \le 2{,}400 \\
& .1ODOOR + .4ODDHEUR + .05MCSMELL \le 600 \\
& ODDHEUR \ge 200 \\
& ODOOR + ODDHEUR + MCSMELL \ge 4{,}000 \\
& ODOOR, \quad ODDHEUR, \quad MCSMELL \ge 0.
\end{aligned}$$

In this model, the first constraint restricts the total use of bergamot, the second constraint limits the use of musk, the third models the bound on the second perfume, and the fourth constrain ensures that the total output exceeds the specified value.

The optimal solution with the options "Display the Optimal Tableau" and "Provide Sensitivity Analyses" is shown below in a simulated printout that is similar to that provided by many commercial packages.

SUMMARY OF RESULTS

VALUE OF THE OBJECTIVE FUNCTION 79,000.00

DECISION VARIABLE	VALUE AT OPTIMUM	OPPORTUNITY COST
ODOOR	5,200.0000	0.0000
ODDHEUR	300.0000	0.0000
MCSMELL	0.0000	3.0000

SLACK/EXCESS VARIABLE	CONSTRAINT TYPE	VALUE AT OPTIMUM	SHADOW PRICE
CONSTRAINT 1:	LE	800.0000	0.0000
CONSTRAINT 2	LE	0.0000	140.0000
CONSTRAINT 3	GE	0.0000	25.0000
CONSTRAINT 4	GE	1,400.0000	0.0000

SENSITIVITY ANALYSES

COEFFICIENTS OF THE OBJECTIVE FUNCTION

VARIABLE	LOWEST ALLOWABLE VALUE	ORIGINAL VALUE	HIGHEST ALLOWABLE VALUE
ODOOR	8.0000	14.0000	INFINITY
ODDHEUR	–INF	31.0000	56.0000
MCSMELL	– INF	4.0000	7.0000

RIGHT-HAND SIDE VALUES

CONSTRAINT NUMBER	LOWEST ALLOWABLE VALUE	ORIGINAL VALUE	HIGHEST ALLOWABLE VALUE
CONSTRAINT 1	1,600.0000	2,400.0000	INFINITY
CONSTRAINT 2	460.0000	600.0000	866.6667
CONSTRAINT 3	0.0000	200.0000	666.6667
CONSTRAINT 4	-INFINITY	4,000.0000	5,400.0000

FINAL TABLEAU

VARIABLE	ODOOR	ODDHEUR	MCSMELL	S1	S2	S3	S4	RHS
CONSTRAINT 1	1.00	0.00	0.50	0.00	10.00	4.00	0.00	5,200
CONSTRAINT 2	0.00	1.00	0.00	0.00	0.00	-1.00	0.00	200
CONSTRAINT 3	0.00	0.00	-0.10	1.00	-3.00	-1.00	0.00	800
CONSTRAINT 4	0.00	0.00	-0.50	0.00	10.00	3.00	1.00	1,400
OBJECTIVE FUNCTION	0.00	0.00	3.00	0.00	140.00	25.00	0.00	79,000

We will now discuss some of the many issues that can be addressed based on the information provided in the printout. In particular, we will state the question followed by two answers: one to an analyst, who is interested in where to find and how to calculate the information, and the decision maker, whose interest concerns the managerial implications.

Q1: What is the solution and how much will it net us?

> *A1 (to the analyst): We are looking for the values of the decision variables and the value of the objective function. The required information is found in the SUMMARY OF RESULTS as well as in the final tableau. Since the latter is typically very large and only supplied on demand, the concise summary of results is an easier source.*
>
> *A1 (to the decision maker): The optimal solution calls for the manufacture and sales of 5,200 bottles of ODOOR, 200 bottles of ODDHEUR, but no bottles of MCSMELL. Associated with this solution is a profit of $79,000.*

Q2: How much of each of the ingredients is used and where are the bottlenecks in our production?

> *A2 (to the analyst): The usage of the first two ingredients—the third resource is assumed to be ubiquitous—is expressed by the first two constraints. All pertinent information can be found in the summary of results. We can either take the values of the three decision variables, insert them into the left-hand sides of the first two constraints and calculate the resource usages that way, or we simply subtract the number of unsused units (the appropriate slack) from the number of units that are available. For bergamot essence, we have originally 2,400 units and the*

slack in the first constraint equals 800, so that we have used 1,600 fl. oz. in the process. Similarly, we start out with 600 fl. oz. of musk and none of it is left over. This makes musk a bottleneck in the process. We also notice that the excess variable of constraint 3 equals zero, indicating that this constraint is also tight, while constraint 4 shows an excess of 1,400 units.

A2 (to the decision maker): We are presently using all of the musk and we are making the exact number of bottles of ODDHEUR that are requested. Hence the limitations on the availability of musk and the requirement to make at least 200 bottles of ODDHEUR are holding us back.

Q3: Let's talk bergamot essence. If we could get some more, how much more would we pay per fl. oz. of additional bergamot essence?

A3 (to the analyst): The question concerns a sensitivity analysis on a right-hand side value. Since the slack variable associated with this resource is positive, it is not a bottleneck (as already elaborated in the previous question) and thus we would not buy any additional quantities. Formally, the same answer is provided by the shadow price, which, for the fist constraint, equals zero.

A3 (to the decision maker): Presently, we have 800 fl. oz. of bergamot left over, so there is no need to purchase any more. However, if we can get a good deal on bergamot today, we may use it in our production in the next planning period.

Q4: How about the musk? Is it worth getting any more of it? And if so, how much would we be prepared to pay for additional quantities? You know, I have this offer for 200 fl. oz. of musk, but they are asking for $110 per fl. oz. It sounds outrageous to me, but hey, that's the spot market. What should I do?

A4 (to the analyst): Again, the question deals with a sensitivity analysis on a right-hand side value. However, in contrast to the previous question, the slack variable that belongs to this resource equals zero, indicating that it is fully used up. The shadow price of this resource is $140, meaning that for each additional fl. oz. of musk that were to be available to us, the profit would increase by $140. The part of the printout that is labeled "Right-hand side values" under the header "Sensitivity Analyses" shows that this result holds true as long as musk is available in quantities between 460 and 866.67 fl. oz. Outside of these bounds, the basis will change and additional calculations

are required to determine the new optimal solution and its profit. Having originally 600 fl. oz. of musk available, an additional 200 fl. oz. will not change the basis and each ounce will result in a profit increase of (shadow price of musk) – (money demanded per ounce) = 140 – 110 = $30. However, since the constraint is binding at optimum, while the basis may not change, the (managerial) solution will change for any change of the availability of musk, regardless how small.

In order to determine what happens if some additional Δb_2 = 200 fl. oz. of musk were offered to us, we either resolve the problem with the new parameters or reconstruct the solution with the help of the final tableau provided (as an option) on the printout. Updating the right-hand side in the final tableau by the column under the slack variable associated with the resource called musk (i.e., S_2), we obtain

$$\begin{bmatrix} z \\ ODOOR \\ ODDHEUR \\ S_1 \\ E_2 \end{bmatrix} = \begin{bmatrix} 79{,}000 + 140\Delta b_2 \\ 5{,}200 + 10\Delta b_2 \\ 200 + 0\Delta b_2 \\ 800 - 3\Delta b_2 \\ 1{,}400 + 10\Delta b_2 \end{bmatrix} = \begin{bmatrix} 107{,}000 \\ 7{,}200 \\ 200 \\ 200 \\ 3{,}400 \end{bmatrix}.$$

In other words, the new solution includes an additional 2,000 bottles of ODOOR, while the number of bottles of ODDHEUR and MCSMELL does not change. Given this change, we have now reduced our resources of bergamot essence to 200 fl. oz., while musk is still a bottleneck. The new profit is $107,000 – (200) (110) = 85,000.

A4 (to decision maker): Musk is one of the bottlenecks in our production. Actually, you could pay up to $140 per fl. oz. and still come out ahead. As a matter of fact, you could get up to 266⅔ fl. oz. for that price, each ounce netting you an extra $30. Beyond that, I'd have to do some more number crunching. In the long run, though, we should stock up on musk oil, as it is usually available for $80 per fl. oz.

Q5: All right, forget about the ingredients. Instead, I want you to focus on our mass product *MCSMELL*. We are favoring the idea of possibly not making much profit per unit, but having mass appeal and make our profit by selling as many units as possible and *MCSMELL* is appealing in that context. It is disappointing that it is not included in the solution. What would it take—as far as its price is concerned—to include the product in the solution?

A5 (to the analyst): The question concerns the sensitivity analysis of on the objective function coefficient of a variable that is nonbasic in the optimal solution. There are two ways to find the answer. The first is to look in the summary of results and determine the opportunity cost of MCSMELL, which happens to be $3. This means that the present price of the bottle of $15 is too low to include the perfume in our production. More specifically, its price is $3 short of the product becoming profitable. The same result can be found in the sensitivity analyses on the objective function coefficients on the printout. There, the unit profit range is from $-\infty$ to 7, which is a $3 increase over the present per-unit profit contribution of $4.

A5 (to the decision maker): At present, MCSMELL is not profitable. however, if we were to raise its price by 20% from $15 to $18 (or more), it will become profitable. Whether or not our customers are prepared to pay that much is something I don't know. If you are serious about this possibility, let me know and I will do some more calculations.

Q6: I wonder what would happen if we were to change the price of our popular brand *ODOOR*. Can you tell me something about that?

A6 (to the analyst): We are talking about the sensitivity analysis of an objective function coefficient. This time, however, the corresponding decision variable is in the basis. Checking the sensitivity analyses in case of changes of coefficients of the objective function, we see that the original optimal solution remains unchanged as long as the unit profit, which is originally $14, remains between $8 and ∞. The upper bound is easily explained: Given the original unit profit contribution of $14, we are making 5,200 bottles. In other words, $14 makes it worth our while to manufacture the product. If its profitability increases, there is no reason for us to change making it. However, if its profitability were to decrease, there comes a point, at which it is no longer worthwhile to make the product and our production plan would change. This happens when the unit profit equals $8, i.e., a decrease of $6 from the original $14.

A6 (to the decision maker): As far as increasing the price of ODOOR is concerned, it will remain profitable. Whether or not our customers will continue to demand ODOOR in the quantities they have until now is something the marketing department will have to find out. As far as price decreases are concerned I can tell you that we should not change our

production plan as long as the prices decreases by no more than $6, i.e., as long as the sales price is $34 or above.

Q7: I have long since been toying with the idea to drop the requirement to make at least 200 bottles of *ODDHEUR*. This will get us out of the specialty market, but maybe that's where we are headed anyway. What happens if we drop this requirement?

A7 (to the analyst): It is apparent that if we drop a constraint, the objective value cannot get worse and it usually improves. From a technical point of view, dropping a constraint can be accomplished by making it redundant. In this model, we achieve this by reducing the right-hand side value of the third constraint to zero, i.e., $\Delta b_3 = -200$. From the tableau we obtain the updated right-hand side value as

$$\begin{bmatrix} z \\ ODOOR \\ ODDHEUR \\ S_1 \\ E_2 \end{bmatrix} = \begin{bmatrix} 79{,}000 - 25\Delta b_3 \\ 5{,}200 - 4\Delta b_3 \\ 200 + 1\Delta b_3 \\ 800 + 1\Delta b_3 \\ 1{,}400 - 3\Delta b_3 \end{bmatrix} = \begin{bmatrix} 84{,}000 \\ 6{,}000 \\ 0 \\ 600 \\ 2{,}000 \end{bmatrix},$$

meaning that without the lower bound on the number of bottles of ODDHEUR, we will be making 6,000 bottles of ODOOR and nothing of the others for a total profit of $84,000, up from the $79,000 that we had originally.

A7 (to the decision maker): As you suspected, without the requirement to make at least 200 bottles of ODDHEUR, we will not make that perfume at all. Actually, we will make only ODOOR, and 6,000 bottles of it. The profit is then $84,000, up by $5,000. That means that each bottle of ODDHEUR that we make decreases our profit by $25. We'd be much better off without it. However, we do have to consider the dangers involved in being a one-perfume company.

Q8: I understand that. Let's get back to *MCSMELL*. As you know it is a mass market product and I would like to make it even more so. Right now it consists of 5% bergamot essence, 5% musk oil, and 90% fill, i.e., oil. As you know, bergamot and musk are expensive and oil is cheap. Can we get away with reducing the content of bergamot and musk? In order for the perfume to maintain its distinctive smell, we have to have equal quantities of bergamot and musk. I don't want to change the price of the product, so how can we change the composition of *MCSMELL* so as to make it profitable?

A8 (to the analyst): Denote by y the quantity of each, bergamot and musk, in each ounce of MCSMELL. The unit profit is then the sales price minus the costs, i.e., $15 - 50y - 80y - 5(1 - 2y) = 10 - 120y$. The opportunity costs of MCSMELL are \$3, *meaning that the unit profit that is presently \$4, has to be at least \$3 higher, i.e., \$7. In other words, it is necessary that* $10 - 120y \geq 7$ *or* $y \leq 0.025$, *i.e., half of the present content of bergamot and musk. If the content of the perfume drops below that level, the solution changes. For instance, if we include 0.024 fl. oz. of each, bergamot and musk, in MCSMELL and top it up with oil, the unit cost of a bottle of MCSMELL are \$7.88, so that its unit profit is \$7.12. Also, the coefficients of MCSMELL in the first two constraints change from 0.05 to 0.024 and the new solution is obtained by resolving the problem. The new solution includes no bottles of ODOOR, 200 bottles of ODDHEUR, and 21,666.67 bottles of MCSMELL with a profit of \$160,466.70. We can now delete the lower bound on ODDHEUR and resolve the problem again, which will improve the solution to no ODOOR, no ODDHEUR, and 25,000 bottles of MCSMELL. The profit in this case is \$178,000.*

A8 (to the decision maker): If we cut the present content of bergamot and musk in half (or even more), MCSMELL becomes profitable. As an example, if we were to use 0.024, 0.024, and 0.952 fl. oz. of bergamot, musk, and oil, respectively, the unit profit of MCSMELL is \$7.12. If we do that, our production plan will include no ODOOR, the mandatory 200 bottles of ODDHEUR, and 21,666.67 bottles of MCSMELL, a dramatic change from the previous solution. Also, the new profit has increased by more than 100% to \$160,466.70, certainly a strong incentive to cheapen MCSMELL by changing its composition as indicated. As a matter of fact, if we were to drop the restriction that we have to make at least 200 bottles of ODDHEUR, the optimal plan is to make 25,000 bottles of MCSMELL and nothing else, thus having completely entered the budget market and left the specialty niche. The profit would then be \$178,000, another 11% increase.

Final words by the decision maker: Thanks so much, Mr. Operations Researcher. I think you have given great decision support, for which I will recommend you for a promotion.

7 NON-SIMPLEX BASED SOLUTION METHODS

So far, we have concentrated on the simplex method as the solution technique of choice for linear programming problems. While being extremely successful in practice for the last half century, the simplex method is by no means the only solution method for linear programming. Interestingly enough, the first alternative solution technique to the simplex method, a method by Brown and Koopmans (1951), was published in the same volume in which Dantzig presented his simplex method in 1951.

Since then, a variety of methods have been suggested, none of which were able to compete with the simplex method as far as computational speed and general practicability was concerned. This changed in 1984, when Karmarkar (1984) proposed a technique that has turned out efficient not only in the worst case, but is also able to solve large-scale methods within a reasonable amount of time. This section will review some of the non-simplex techniques that have been put forward over the last 70 years.

From a formal point of view, consider a linear programming problem in canonical form, which is, as usual, written in matrix notation as

$$\begin{aligned} \text{P: Max } z &= \mathbf{cx} \\ \text{s.t.} \quad & \mathbf{Ax} \leq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}. \end{aligned}$$

The dual of problem P is then

$$\begin{aligned} \text{P}_\text{D}\text{: Min } z_D &= \mathbf{ub} \\ \text{s.t.} \quad & \mathbf{uA} \geq \mathbf{c} \\ & \mathbf{u} \geq \mathbf{0}. \end{aligned}$$

Results from duality theory (see Corollary 4.12) then assert that any optimal solution ($\mathbf{x}$, $\mathbf{u}$) will satisfy the system of linear (in-) equalities

$$
\begin{aligned}
\text{P*: } \mathbf{Ax} &\le \mathbf{b} \\
\mathbf{x} &\ge \mathbf{0} \\
\mathbf{uA} &\ge \mathbf{c} \\
\mathbf{u} &\ge \mathbf{0} \\
\mathbf{cx} &= \mathbf{ub}
\end{aligned}
$$

Any solution method can then either solve the problem P, its dual P_D, or the system P*. Whatever strategy the solution method uses, the primal and dual solutions are either available directly or can easily be computed via complementary slackness results.

7.1 Alternatives to the Simplex Method

The idea of the *traversal method* by Brown and Koopmans (1951) is fairly simple. It works directly on the (primal) problem. It starts with some interior point $\mathbf{x}^0$ and moves in the direction of $\mathbf{c}$, i.e., the gradient of the objective function, through the interior of the feasible set until it reaches its boundary at some point $\mathbf{x}'$ on some hyperplane H_i. The method then determines a point $\mathbf{x}''$ with $\mathbf{cx}'' = \mathbf{cx}'$, such that $\mathbf{x}''$ is located on a different hyperplane H_k, $k \neq i$. If the points $\mathbf{x}'$ and $\mathbf{x}''$ are not already optimal, such a point always exists. It is apparent that in three or more dimensions, the point $\mathbf{x}''$ will not be unique. A new interior point $\mathbf{x}^1$ is then determined as a linear convex combination of $\mathbf{x}'$ and $\mathbf{x}''$, i.e., $\mathbf{x}^1 = \lambda\, \mathbf{x}' + (1-\lambda)\, \mathbf{x}''$. The search then continues with this point and the procedure is repeated.

The procedure will be illustrated by the following

Example: Consider the linear programming problem

$$
\begin{aligned}
\text{P: Max } & 3x_1 + x_2 \\
\text{s.t. } & 2x_1 + x_2 \le 5 && (I) \\
& x_2 \le 3 && (II) \\
& x_1 - x_2 \le 1 && (III) \\
& x_1,\ x_2 \ge 0.
\end{aligned}
$$

Suppose that the initial feasible solution is $\mathbf{x}^0 = (⅔, 2)$ with $\mathbf{cx}^0 = 4$. Moving along the gradient of the objective function leads to the point

$$
\mathbf{x}' = \mathbf{x}^0 + s\mathbf{c} = (⅔, 2)^T + s(3, 1)^T = \begin{pmatrix} \frac{2}{3} + 3s \\ 2 + 1s \end{pmatrix}
$$

with step length s. Since the new point must be feasible, we require that

$2(⅔+3s) + (2 + s) \le 5$ or $s \le 5/21$	(via constraint *I*)
$2 + s \le 3$ or $s \le 1$	(via constraint *II*)
$(⅔ + 3s) - (2 + s) \le 1$ or $s \le 7/6$	(via constraint *III*)
$(⅔ + 3s) \ge 0$ or $s \ge -2/9$	(via $x_1 \ge 0$), and
$(2 + s) \ge 0$ or $s \ge -2$	(via $x_2 \ge 0$).

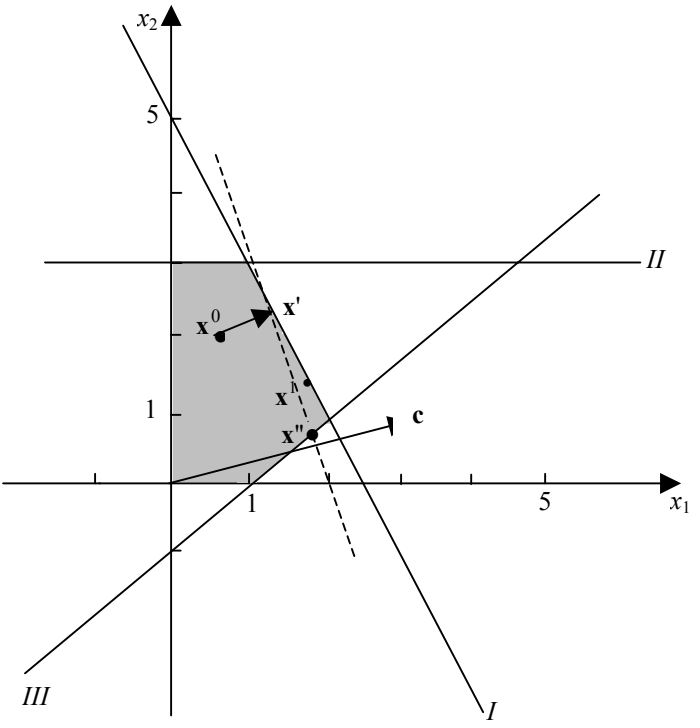


Figure 7.1

The largest possible value of s is then $\frac{5}{21}$, so that the point $\mathbf{x}' = \begin{pmatrix} \frac{2}{3}+3s \\ 2+1s \end{pmatrix} = \begin{pmatrix} \frac{29}{21} \\ \frac{47}{21} \end{pmatrix} \approx \begin{pmatrix} 1.3810 \\ 2.2381 \end{pmatrix}$ is located on hyperplane *I*, i.e., constraint *I* is binding at $\mathbf{x}'$. The objective value at this point is $\mathbf{cx}' = 6\frac{8}{21}$. Determining the point $\mathbf{x}''$ usually allows many degrees of freedom. However, in two dimensions, this point is uniquely determined as $\mathbf{x}'' = \left(\frac{155}{84}, \frac{71}{84}\right) \approx (1.8452, .8452)$, at which point the constraint *III* is binding. The next interior point is then some linear convex combination of $\mathbf{x}'$ and $\mathbf{x}''$, e.g., $\mathbf{x}^1 = ½\ \mathbf{x}' + ½\ \mathbf{x}'' = (271/168, 259/168)^T \approx (1.6131, 1.5417)^T$. The method

will then proceed at this point. The first step is shown in Figure 7.1. Like many of its successors, the method is plagued by slow convergence, particularly when the shapes of the feasible sets are "long and narrow." In such cases, successive points will be located on a path that is zigzagging in a narrow cone that is part of the feasible set.

Another attempt that allows movements through the interior of the feasible set was first suggested by DeMarr (1983) and later elaborated upon by Eiselt and Sandblom (1985, 1990). Interestingly, the *external pivoting method* "tricks" the simplex method into moving through the interior even though the method is designed to follow the boundary of the feasible set. As such, it can either be applied to the primal or the dual problem. This is achieved by introducing an additional "external" variable x^{ext} into the problem and create new "internal" variables x'_j, so that the original variables are expressed as the sum of the new internal and the external variable. In particular, we let $x_j = x'_j + w_j x^{ext}$, with weights $w_j \geq 0 \ \forall \ j$. Clearly, there is a large degree of freedom for choosing the weights. Observe that by introducing the external variable into the basis, all variables x_j for which the weight $w_j > 0$, will have their values increased simultaneously. More specifically, if the external variable were to increase from its present value of zero to, say, one, then all variables whose columns were used to generate the external column increase by w_j. In other words, this method allows us to increase the value of the variables for which positive weights have been chosen simultaneously, so that their values increase in the same ratio to each other as their weights. Since the ratios between the optimal values are unknown (if they were known, we could move to the optimal solution in a single step), one possibility is to approximate them by using $w_j = c_j \ \forall \ j$, i.e., in the direction of the gradient.

Formally, the linear programming in canonical form

$$\begin{aligned} \text{P: Max } z &= \mathbf{cx} \\ \text{s.t.} \quad \mathbf{Ax} &\leq \mathbf{b} \\ \mathbf{x} &\geq \mathbf{0} \end{aligned}$$

has been transformed into the problem

$$\begin{aligned} \text{P': Max } z' &= \mathbf{c}(\mathbf{x}' + \mathbf{w}x^{\text{ext}}) \\ \text{s.t.} \quad \mathbf{A}(\mathbf{x}' + \mathbf{w}x^{\text{ext}}) &\leq \mathbf{b} \\ \mathbf{x}' + \mathbf{w}x^{\text{ext}} &\geq \mathbf{0}, \end{aligned}$$

or simply

$$\begin{aligned} \text{P': Max } z' &= \mathbf{cx}' + \mathbf{cw}x^{\text{ext}} \\ \text{s.t.} \quad \mathbf{Ax}' + \mathbf{Aw}x^{\text{ext}} &\leq \mathbf{b} \\ \mathbf{x}' + \mathbf{w}x^{\text{ext}} &\geq \mathbf{0}. \end{aligned}$$

Clearly, the column of the external variable is a linear combinations of the generating internal columns, i.e., those that belong to variables with positive weights. For simplicity, we replace the nonnegativity constraints in the problem P' by the stronger individual constraints $\mathbf{x}' \geq \mathbf{0}$ and $x^{\text{ext}} \geq 0$. This does not cause problems, as each solution—including the optimal solution—can be expressed by at least one solution that satisfies these stronger constraints as well.

The advantage of this approach is that it enables the user to move through the interior of the feasible set to its boundary, rather than move along its boundaries, thus potentially saving computational effort. Furthermore, the method can very easily be incorporated in the existing software that uses the simplex method. The disadvantage is that, due to its construction, the external column will create some degree of dual degeneracy.

Example: Consider the same linear programming problem that was used to illustrate the Brown-Koopmans approach. Its first tableau is

T^1:

x_1	x_2	S_1	S_2	S_3	1
2	1	1	0	0	5
0	2	0	1	0	3
1	–1	0	0	1	1
–3	–1	0	0	0	0

Generating the external column with arbitrarily chosen weights $\mathbf{w} = (4, 1)$, we obtain the augmented tableau

$T^{1\prime}$:

x_1'	x_2'	x^{ext}	S_1	S_2	S_3	1
2	1	9	1	0	0	5
0	2	2	0	1	0	3
1	–1	(3)	0	0	1	1
–3	–1	–13	0	0	0	0

Introducing the external column into the basis results in tableau

T^2:

x_1'	x_2'	x^{ext}	S_1	S_2	S_3	1
–1	(4)	0	1	0	–3	2
$-\frac{2}{3}$	$\frac{8}{3}$	0	0	1	$-\frac{2}{3}$	2⅓
$\frac{1}{3}$	$-\frac{1}{3}$	1	0	0	$\frac{1}{3}$	$\frac{1}{3}$
$\frac{4}{3}$	$-\frac{16}{3}$	0	0	0	$\frac{13}{3}$	4⅓

If desired, we could reconstruct the values of the original variables as $x_1 = x_1' + 4x^{ext} = 0 + 4(\frac{1}{3}) = \frac{4}{3}$ and $x_2 = x_2' + 1x^{ext} = 0 + \frac{1}{3} = \frac{1}{3}$. As T^2 is not yet optimal, another iteration introduces x_2' into the basis. The tableau is then

T^3:

x_1'	x_2'	x^{ext}	S_1	S_2	S_3	1
$-\frac{1}{4}$	1	0	$\frac{1}{4}$	0	$-\frac{3}{4}$	$\frac{1}{2}$
0	0	0	$-\frac{2}{3}$	1	$\frac{4}{3}$	1
$\frac{1}{4}$	0	1	$\frac{1}{12}$	0	$\frac{1}{12}$	$\frac{1}{2}$
0	0	0	$\frac{4}{3}$	0	$\frac{1}{3}$	7

This tableau is optimal, and the optimal solution can be determined as $\bar{x}_1 = x_1' + 4x^{ext} = 0 + 4(½) = 2$ and $\bar{x}_2 = x_2' + 1x^{ext} = ½ + ½ = 1$, so that $\bar{z} = \mathbf{c}\bar{\mathbf{x}} = 7$.

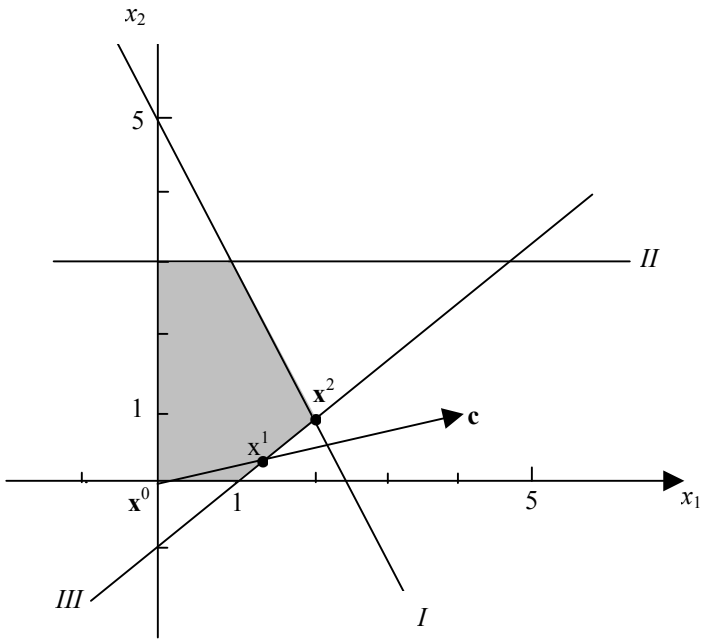


Figure 7.2

Note the dual degeneracy shown by the zero indicator under the variable x_1'. This signifies that alternative optimal solutions exist. Introducing the variable x_1' into

the basis, the next tableau shows that $x_1' = 2$, $x_2' = 1$, and $x^{\text{ext}} = 0$, so that $\bar{x}_1 = 2$ and $\bar{x}_2 = 1$, formally a different solution, even though the values of the original variables are the same. Figure 7.2 shows the progression of the solutions from the original solution to the optimum.

A different approach was taken by Murty (1986). His *gravitational method* is modeled after a physical equivalent, using an idea from Fourier in the 1820s. Again, the method works directly on the primal problem under consideration. In particular, it first rotates the space of decision variables, so that the gradient points directly downwards. (Actually, it is not necessary to perform such a rotation, it is useful only for explanatory purposes). The method starts with an interior solution, which is thought of as a ball of a liquid that follows the path of gravity. This means that the ball will first fall onto one of the hyperplanes that bounds the feasible set, and from that point on, it will follow a path along the hyperplanes that define the feasible set to the lowest point. This point is the optimal solution. Testing of this method on real-world problems is not available.

A graphical representation of the gravitational method is shown in Figure 7.3. The initial solution is again $\mathbf{x}^0$, from where the ball of liquid drops to $\mathbf{x}^1$, from where it follows the boundary of the feasible set down to the optimal solution at $\mathbf{x}^2$.

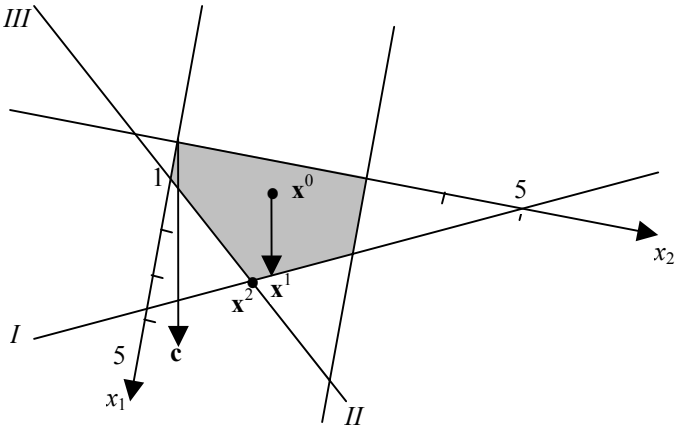


Figure 7.3

Another technique that is also based on a physical equivalent was proposed by Eiselt and Sandblom (2000a). Their *bounce method* mimics the path of a ball that bounces off the inside of the surfaces that define the feasible set. Ignoring friction, the ball will also come to rest at the lowest point of the feasible set, which is the optimum. One may view this as an extension of Murty's gravitation method, given that the ball does not bounce but only slides down the surfaces.

The *ellipsoid method*, proposed by Khachian (1979) made quite a stir in the scientific community. Its major contribution was that, for the first time, it was proved that it was possible to solve linear programming problems in polynomial time in the worst case. However, once it became implemented, it soon turned out that the method, while making a significant theoretical contribution by performing efficiently in the worst case, did not perform well on average and thus was not suitable for the solution of linear programming problems in practice. Still, based on its landmark contribution, we will briefly describe how the ellipsoid method works. The ellipsoid method is designed to find feasible solutions to a set of simultaneous linear equations efficiently, so it will have to be applied to the problem P* as defined at the beginning of this chapter. Note that P* has either at least one feasible solution, or no feasible solutions. If, say, the primal problem P has unbounded "optimal" solutions, then the dual P_D will have no feasible solutions, so that P* will also have no feasible solutions.

This system can be suitably modified, so that it is equivalent to solving a system of strict linear inequalities. For simplicity of notation, this system will be referred to as $\mathbf{Ax} < \mathbf{b}$ and the problem is solved, if a solution is found to this system. Assume that $\mathbf{A}$ is an $[m \times n]$-dimensional matrix and let all parameters a_{ij} and b_i be integers. We initialize the method with the point $\mathbf{x}^0 = [0, 0, \ldots, 0]^T$ and a hypersphere centered at $\mathbf{x}^0$ that is sufficiently large so as to include at least one of the feasible points of the system (if at least one feasible point exists). Suppose now that the initial point $\mathbf{x}^0$ is not feasible, otherwise the task of finding a feasible solution has been accomplished and the algorithm terminates. Set now the iteration counter k:= 0 and let the present ellipsoid—initially the hypersphere—be E^k.

Since the present solution $\mathbf{x}^k$ is not feasible by assumption, there exists at least one violated constraint, say $\mathbf{a}_i \bullet \mathbf{x}^k \geq b_i$, meaning that $\mathbf{x}^k$ is located on the "wrong side" of hyperplane H_i. We now shift the hyperplane in parallel fashion, so that it leads through $\mathbf{x}^k$. This process results in the hyperplane H_i', which cuts the current ellipsoid E^k into two parts: one part, say, E_+^k that includes all feasible points in E^k (if any), and the other part is E_-^k, which does not include any feasible points. Let now H_i'' be a hyperplane that is parallel to H_i, which is tangent to E^k. Then a new ellipsoid E^{k+1} with center $\mathbf{x}^{k+1}$ is constructed, such that the hull of E^{k+1} includes the intersection of H_i' with E^k and the intersection of H_i'' with E_+^k (note that $E^{k+1} \supset E_+^k$). Then $\mathbf{x}^{k+1}$ is the new point, and the procedure is repeated if necessary. Figure 7.4 may explain the procedure.

The shaded area in Figure 7.4, bounded by the hyperplanes H_1, H_2, and H_3 represents the set of feasible solutions, the origin is the initial point $\mathbf{x}^0$, and the circle E^0 is the initial ellipsoid. Since $\mathbf{x}^0$ satisfies the constraints that belong to H_2 and H_3, but violates the constraint represented by H_1, the hyperplane H_1 is shifted

in parallel fashion through $\mathbf{x}^0$, thus cutting E^0 into two semicircles; E_+^0 northeast of H_1' and E_-^0 southwest of H_1'. The intersections of H_1' and E^0 are the points A and B. We now shift the hyperplane H_1 in parallel fashion in a northeasterly direction until it is tangent to E^0, this hyperplane is H_1'' and it touches the ellipsoid E^0 at the point C. The new ellipsoid E^1 with its center at $\mathbf{x}^1$ is then constructed, so that it includes the points A, B, and C on its hull and it completely circumscribes the ellipsoid E_+^0.

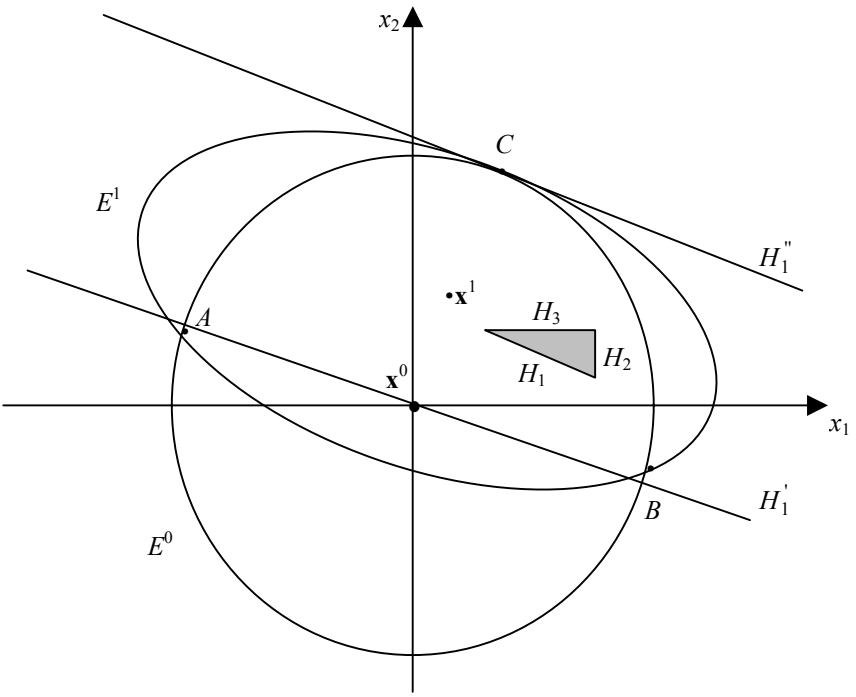


Figure 7.4

The next step will examine the point $\mathbf{x}^1$. Since this point satisfies the first two constraints but violates the third, the hyperplane H_3 will be shifted through $\mathbf{x}^1$, cutting the ellipsoid E^1 into two half ellipsoids and the procedure is repeated.

The main two issues are the size of the initial ellipsoid and the rate at which the ellipsoids are shrinking. Clearly, while the initial ellipsoid must be sufficiently large so as to include at least one feasible solution, it is desirable that it is as small as possible, so as to speed up convergence. In order to determine the size of the initial hypersphere, we need to determine the size of the binary encoding of the problem. Given the $[m \times n]$-dimensional matrix $\mathbf{A}$ and the m-dimensional column vector $\mathbf{b}$, the length of the encoding is

$$L = \sum_{i=1}^{m}\sum_{j=1}^{n}\lfloor \log_2 | a_{ij} | \rfloor + \sum_{i=1}^{m}\lfloor \log_2 | b_i | \rfloor + \lfloor \log_2 n \rfloor + \lfloor \log_2 m \rfloor + 2m(n+1) + 4 .$$

This expression derives from the need to encode the size of the problem, all coefficients in the matrix **A** including the signs, and the right-hand side vector **b.** All coefficients are separated by signs. Specifically, let each symbol be either +, –, 0, or 1. Consider now some given integer α. If α = 0, then a single symbol is required for its encoding. Otherwise, we need $\lfloor \log_2 \alpha \rfloor$ symbols to express the number plus an extra symbol for the sign. The problem P* can then be encoded by first stating the numbers m and n (requiring $\lfloor \log_2 m \rfloor + 2$ and $\lfloor \log_2 n \rfloor + 2$ symbols including the signs, respectively), the coefficients of the matrix **A** require $\sum_i \sum_j \lfloor \log_2 | a_{ij} | \rfloor$ symbols (excluding the signs), and the vector **b** necessitates $\sum_i 1 + \lfloor \log_2 | b_i | \rfloor$ symbols, again, excluding the signs.

It is then possible to prove that a hypersphere centered at the origin and with radius 2^L includes at least one feasible point, provided such a point exists. For details, see Khachian (1979), or Nemhauser and Wolsey (1988). Unfortunately, this initial ellipsoid is extremely large even for small problems. This feature, coupled with a slow rate of shrinkage of the ellipsoids, is responsible for the exceedingly slow rate of convergence of the ellipsoid method.

However, the method will converge in a number of steps that is polynomial in the size of the problem, making it an efficient method in the worst case. Formally, we can state

Lemma 7.1: The ellipsoid method terminates after at most $6n(n+1)L$ iterations with either a feasible solution to the problem P or with an indication that no feasible solution exists.

This lemma is used as a stop criterion in the formal description of the method below. We initialize the method by setting $\mathbf{x}^0 := [0, 0, \ldots, 0]^T$, $k := 0$, and defining an $[n \times n]$-dimensional matrix $\mathbf{D}^0 = 2^L\mathbf{I}$. Furthermore, we assume that all calculations can be carried out with infinite precision, i.e., given exact calculations. We can then formally state the ellipsoid method.

The Ellipsoid Method

Step 1: Is $\mathbf{x}^k$ a feasible solution to the problem P*?

If yes: Stop, $\mathbf{x}^k$ is a solution to the problem P*.
If no: Go to Step 2.

Step 2: Is $k > 6n(n+1)L$?
If yes: Stop, no feasible solution to the problem P exists.
If no: Go to Step 3.

Step 3: Suppose that the i-th constraint is violated, i.e., $\mathbf{a}_{i\bullet}\mathbf{x}^k \geq b_i$. Then determine

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \frac{1}{n+1}\frac{\mathbf{D}^k\mathbf{a}_{i\bullet}^T}{\sqrt{\mathbf{a}_{i\bullet}\mathbf{D}^k\mathbf{a}_{i\bullet}^T}} \text{ and}$$

$$\mathbf{D}^{k+1} = \frac{n^2}{n^2-1}\left[\mathbf{D}^k - \frac{2}{n+1}\frac{\left[\mathbf{D}^k\mathbf{a}_{i\bullet}^T\right]\left[\mathbf{D}^k\mathbf{a}_{i\bullet}^T\right]^T}{\mathbf{a}_{i\bullet}\mathbf{D}^k\mathbf{a}_{i\bullet}^T}\right].$$

Set $k := k + 1$ and go to Step 1.

Note that the numerator in the last ratio of the second formula is the product of a column vector and a row vector, i.e., a matrix.

The ellipsoid method can now be illustrated in the following

Example: Consider the system of simultaneous strict inequalities

$$\begin{aligned} \text{P: } & x_1 - x_2 < -1 \\ & -x_1 \quad\; < -1, \end{aligned}$$

i.e., $\mathbf{a}_{1\bullet} = [1, -1]$, $\mathbf{a}_{2\bullet} = [-1, 0]$, $b_1 = -1$, $b_2 = -1$, $n = 2$, $m = 2$, and $L = 18$. To accelerate the procedure, we set $L := 5$, knowing that the circle with center at the origin and with radius $2^L = 32$ still includes feasible points, e.g., $x_1 = 2$ and $x_2 = 4$. The initial solution is $\mathbf{x}^0 = [0, 0]^T$ and $\mathbf{D}^0 = \begin{bmatrix} 32 & 0 \\ 0 & 32 \end{bmatrix}$. It is apparent that the initial solution violates both constraints, and we arbitrarily select constraint 1. Then

$$\mathbf{x}^1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \frac{1}{3}\frac{\begin{bmatrix} 32 & 0 \\ 0 & 32 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \end{bmatrix}}{\sqrt{[1,-1]\begin{bmatrix} 32 & 0 \\ 0 & 32 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \end{bmatrix}}} = \begin{bmatrix} -\frac{4}{3} \\ \frac{4}{3} \end{bmatrix} \text{ and}$$

$$\mathbf{D}^1 = \frac{4}{3}\left[\begin{bmatrix}32 & 0\\ 0 & 32\end{bmatrix} - \frac{2}{3}\frac{\left[\begin{bmatrix}32 & 0\\ 0 & 32\end{bmatrix}\begin{bmatrix}1\\ -1\end{bmatrix}\left[\begin{bmatrix}32 & 0\\ 0 & 32\end{bmatrix}\begin{bmatrix}1\\ -1\end{bmatrix}\right]^T\right]}{\begin{bmatrix}1 & -1\end{bmatrix}\begin{bmatrix}32 & 0\\ 0 & 32\end{bmatrix}\begin{bmatrix}1\\ -1\end{bmatrix}}\right]$$

$$= \begin{bmatrix}\frac{256}{9} & \frac{128}{9}\\ \frac{128}{9} & \frac{256}{9}\end{bmatrix}.$$

The solution $\mathbf{x}^1$ now satisfies the first constraint but still violates the second constraint, so that we calculate

$$\mathbf{x}^2 = \begin{bmatrix}-\frac{4}{3}\\ \frac{4}{3}\end{bmatrix} - \frac{1}{3}\frac{\begin{bmatrix}\frac{256}{9} & \frac{128}{9}\\ \frac{128}{9} & \frac{256}{9}\end{bmatrix}\begin{bmatrix}-1\\ 0\end{bmatrix}}{\sqrt{\begin{bmatrix}-1,0\end{bmatrix}\begin{bmatrix}\frac{256}{9} & \frac{128}{9}\\ \frac{128}{9} & \frac{256}{9}\end{bmatrix}\begin{bmatrix}-1\\ 0\end{bmatrix}}} = \begin{bmatrix}\frac{4}{9}\\ \frac{20}{9}\end{bmatrix} \text{ and}$$

$$\mathbf{D}^2 = \frac{4}{3}\left[\begin{bmatrix}\frac{256}{9} & \frac{128}{9}\\ \frac{128}{9} & \frac{256}{9}\end{bmatrix} - \frac{2}{3}\frac{\left[\begin{bmatrix}\frac{256}{9} & \frac{128}{9}\\ \frac{128}{9} & \frac{256}{9}\end{bmatrix}\begin{bmatrix}-1\\ 0\end{bmatrix}\left[\begin{bmatrix}\frac{256}{9} & \frac{128}{9}\\ \frac{128}{9} & \frac{256}{9}\end{bmatrix}\begin{bmatrix}-1\\ 0\end{bmatrix}\right]^T\right]}{\begin{bmatrix}-1, & 0\end{bmatrix}\begin{bmatrix}\frac{256}{9} & \frac{128}{9}\\ \frac{128}{9} & \frac{256}{9}\end{bmatrix}\begin{bmatrix}-1\\ 0\end{bmatrix}}\right]$$

$$= \begin{bmatrix}\frac{1,024}{81} & \frac{512}{81}\\ \frac{512}{81} & \frac{2,560}{81}\end{bmatrix}.$$

The new solution $\mathbf{x}^2$ still violates the second constraint, hence we compute

$$\mathbf{x}^3 = \begin{bmatrix}\frac{4}{9}\\ \frac{20}{9}\end{bmatrix} - \frac{1}{3}\frac{\begin{bmatrix}\frac{1,024}{81} & \frac{512}{81}\\ \frac{512}{81} & \frac{2,560}{81}\end{bmatrix}\begin{bmatrix}-1\\ 0\end{bmatrix}}{\sqrt{\begin{bmatrix}-1,0\end{bmatrix}\begin{bmatrix}\frac{1,024}{81} & \frac{512}{81}\\ \frac{512}{81} & \frac{2,560}{81}\end{bmatrix}\begin{bmatrix}-1\\ 0\end{bmatrix}}} = \begin{bmatrix}\frac{44}{27}\\ \frac{76}{27}\end{bmatrix} \approx \begin{bmatrix}1.630\\ 2.815\end{bmatrix},$$

which is a feasible solution, so that $\mathbf{D}^3$ does not have to be computed and the procedure terminates.

7.2 Interior Point Methods

Although the simplex method is still the dominant procedure for solving linear programming problems, it has a major shortcoming: it cannot be guaranteed to find an optimal solution in polynomial time. On average, a problem with m structural constraints will require ½(3m) iterations until an optimal solution is reached. For a discussion and references, see Vanderbei (2001), who specifies the number as ½(m + n), which for a square problem with $m = n$ and with slack variables amounts to ½(3m). Further details can be found in Dantzig and Thapa (2003). While this number may apply on average, there are problems for which many more iterations are required. For example, the examples provided by Klee and Minty (1972), in which the feasible region is a distorted hypercube in $\mathbb{R}^n$, will force the simplex method to visit all 2^n extreme points of the feasible region before reaching optimum. This is the reason why researchers have been trying to find methods that can be proved to exhibit polynomial behavior. Apparently, this was already accomplished by the logarithmic potential method of Frisch (1956), which was further developed by Parisot (1961); the simplex splitting technique due to Levin (1965), as well as the center method of Huard (1967), which was fully developed by Renegar (1988). However, it was the *ellipsoid method* by Khachian (1979) that was the first widely known polynomial-time algorithm for general linear programming problems.

The simplex method was thus unrivalled for the solution of practical linear programming problem, until the modern development of *interior point methods*. They were first rediscovered by Karmarkar (1984), whose *projective scaling method* was able to compete with the simplex method as applied to realistic problems. A remarkable feature of computer implementations of interior point methods is that the number of iterations required by the method does not increase very rapidly with the size of the problem. As a matter of fact, it appears that less than one hundred iterations are sufficient even for the solution of very large problems with millions of variables (realizing, of course, that one iteration with an interior point method is much more complex than one iteration with any of the simplex methods). This is in stark contrast to the number of iterations required by the simplex method. Recall that the simplex method moves along the boundary of the feasible region from one extreme point to an adjacent extreme point. Typically, real-life problems include an astronomical number of extreme points. McMullen (1970) has shown that for a linear programming problem with n variables and m constraints, there could be as many as $\binom{m-\left\lfloor \frac{n+1}{2} \right\rfloor}{m-n} + \binom{m-\left\lfloor \frac{n+2}{2} \right\rfloor}{m-n}$ extreme points. Even for small problems with n and m in the hundreds, this number can easily surpass 10^{100}, a number of extreme points that is far too large to be examined, even if only a tiny fraction would have to be dealt with. It is one of the great achievements of the simplex method that, on average, it is generally agreed that it needs to explore no more than about $\frac{3}{2}m$ of the existing extreme points.

Since that time, the class of interior point methods has been developed to the extent that some commercial software packages for large-scale linear programming now offer interior point methods as alternatives to the simplex method. Interior point methods are treated in a number of books on linear programming. Useful pertinent references are Padberg (1995), Saigal (1995), Rardin (1998), Bhatti (2000), Vanderbei (2001), Sierksma (2002), Dantzig and Thapa (2003), and Roos *et al*. (2006).

As the name suggests, interior point methods approach an optimal point (which we know must always be positioned on the boundary of the feasible set) through a sequence of interior points. Starting with some initial interior point, the method moves through the interior of the feasible set along some improving direction to another interior point. There, a new improving direction is found, along which a move is made to yet another interior point. This process is repeated, resulting in a sequence of interior points that converge to an optimal boundary point. The question is now how to stop short of hitting the boundary when we move in an improving direction. Since we are always staying in the (relative) interior of the feasible region, we can easily at each point recompute the direction for the next move; this would not have been easy to do for boundary and extreme points.

There are essentially two different approaches. One is to rescale the problem in order to make the current point stay some distance away from any boundary constraint and then restrict the step length, so that the next move will not reach the boundary. This approach is used in the *affine scaling method* due to Dikin (1967), which we will describe below. The other possibility is to add a (logarithmic) boundary repulsion term to the objective function which prevents the move from reaching the boundary. This technique is used in what is known as the *Newton step barrier method*, which we will also describe.

Below, we will describe the affine scaling method, which assumes that the linear programming problem P is stated in standard form

P: Max $z = \mathbf{cx}$
s.t. $\mathbf{Ax} = \mathbf{b}$
$\mathbf{x} \geq \mathbf{0}$.

Suppose that in some iteration k, we have an interior point $\mathbf{x}^k$, i.e. a point at which $\mathbf{Ax}^k = \mathbf{b}$ and $\mathbf{x}^k > 0$ (or $x_j^k > 0 \ \forall \ j = 1, \ldots, n$). The first step of the method is to rescale the problem to make the current point equal to the summation vector $\mathbf{e} = [1, 1, \ldots, 1]^T$. This is accomplished by defining a diagonal matrix $\mathbf{T}_k$, whose diagonal elements are x_j^k, i.e.,

$$\mathbf{T}_k = \text{diag}\,(\mathbf{x}^k) = \begin{bmatrix} x_1^k & 0 & \ldots & 0 \\ 0 & x_2^k & \ldots & 0 \\ \vdots & \ldots & \ldots & \vdots \\ 0 & 0 & \cdots & x_n^k \end{bmatrix}.$$

Then we define the scaled variable $\tilde{\mathbf{x}}$ as

$$\tilde{\mathbf{x}} := \mathbf{T}_k^{-1}\mathbf{x} = \begin{bmatrix} x_1 / x_1^k \\ x_2 / x_2^k \\ \vdots \\ x_n / x_n^k \end{bmatrix}, \text{ i.e., } \mathbf{x} = \mathbf{T}_k\tilde{\mathbf{x}}, \text{ so that } \tilde{\mathbf{x}}^k = \mathbf{T}_k^{-1}\mathbf{x}^k = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = \mathbf{e}.$$

We then have $z = \mathbf{cx} = \mathbf{cT}_k\tilde{\mathbf{x}} := \mathbf{c}^k\tilde{\mathbf{x}}$, where $\mathbf{c}^k := \mathbf{cT}_k$. Then the system $\mathbf{Ax} = \mathbf{b}$ can be written as $\mathbf{AT}_k\tilde{\mathbf{x}} = \mathbf{b}$, or $\mathbf{A}_k\tilde{\mathbf{x}} = \mathbf{b}$, where $\mathbf{A}_k := \mathbf{AT}_k$. Since the diagonal elements of $\mathbf{T}_k$ are all strictly positive, $\mathbf{x} \geq \mathbf{0}$ is equivalent to $\tilde{\mathbf{x}} \geq \mathbf{0}$. We can now write the linear programming P in its equivalent scaled form

$$\begin{aligned} \tilde{\text{P}}: \text{Max } z = \ & \mathbf{c}^k\tilde{\mathbf{x}} \\ \text{s.t.} \quad & \mathbf{A}_k\tilde{\mathbf{x}} = \mathbf{b} \\ & \tilde{\mathbf{x}} \geq \mathbf{0}, \end{aligned}$$

where we know the current interior feasible point $\tilde{\mathbf{x}}^k = \mathbf{e}$.

The next step is to move from the current point $\tilde{\mathbf{x}}^k$ in an improving feasible direction. It turns out that this task can be accomplished by projecting the gradient of the objective function $(\mathbf{c}^k)^T$ onto the set $\mathbf{A}_k\tilde{\mathbf{x}} = \mathbf{b}$ by means of the projection matrix $\mathbf{P}_k := \mathbf{I} - \mathbf{A}_k^T\left(\mathbf{A}_k\mathbf{A}_k^T\right)^{-1}\mathbf{A}_k$, where we assume that the matrix $\mathbf{A}_k$ has full rank, so that $\mathbf{A}_k\mathbf{A}_k^T$ is nonsingular. To realize this, we consider what happens once we start from $\tilde{\mathbf{x}}^k$, which belongs to the set $\mathbf{A}\tilde{\mathbf{x}} = \mathbf{b}$ (since $\mathbf{A}\tilde{\mathbf{x}}^k = \mathbf{b}$) and move along the projected gradient $\mathbf{P}_k(\mathbf{c}^k)^T$. Let $\alpha > 0$ be some given scalar and consider a move from $\tilde{\mathbf{x}}^k$ to $\tilde{\mathbf{x}}^k + \alpha\mathbf{P}_k(\mathbf{c}^k)^T$. We then obtain for the new point $\mathbf{A}_k\left(\tilde{\mathbf{x}}^k + \alpha\mathbf{P}_k(\mathbf{c}^k)^T\right) = \mathbf{A}_k\tilde{\mathbf{x}}^k + \alpha\mathbf{A}_k\mathbf{P}_k(\mathbf{c}^k)^T = \mathbf{b} + \alpha\mathbf{A}_k\mathbf{P}_k(\mathbf{c}^k)^T$. Now $\mathbf{A}_k\mathbf{P}_k = \mathbf{A}_k\left(\mathbf{I} - \mathbf{A}_k^T\left(\mathbf{A}_k\mathbf{A}_k^T\right)^{-1}\mathbf{A}_k\right) = \mathbf{A}_k - \mathbf{A}_k = \mathbf{0}$, so that $\mathbf{A}_k\left(\tilde{\mathbf{x}}^k + \alpha\mathbf{P}_k(\mathbf{c}^k)^T\right) = \mathbf{b}$, proving that the new point $\tilde{\mathbf{x}}^k$ is still in the set $\mathbf{A}_k\tilde{\mathbf{x}} = \mathbf{b}$. It can also be shown that for $\alpha > 0$, the new point has an improved value of the objective function. In order to show this, we need the following two properties that define the matrix $\mathbf{P}_k$ as a projection matrix:

- $\mathbf{P}_k^T = \mathbf{P}_k$ (symmetry), and
- $\mathbf{P}_k^2 = \mathbf{P}_k$ (idempotence), which are shown in the Appendix at the end of this chapter.

Using these properties, we find that $\mathbf{c}^k\left(\tilde{\mathbf{x}}^k + \alpha \mathbf{P}_k (\mathbf{c}^k)^T\right) = \mathbf{c}^k \tilde{\mathbf{x}}^k + \alpha \mathbf{c}^k \mathbf{P}_k (\mathbf{c}^k)^T$, but

$$\begin{aligned}
&\mathbf{c}^k \mathbf{P}_k (\mathbf{c}^k)^T = \\
&= \mathbf{c}^k \mathbf{P}_k \mathbf{P}_k (\mathbf{c}^k)^T \text{ (due to the idempotence of } \mathbf{P}_k) \\
&= \mathbf{c}^k \mathbf{P}_k \mathbf{P}_k^T (\mathbf{c}^k)^T \text{ (due to the symmetry of } \mathbf{P}_k) \\
&= (\mathbf{c}^k \mathbf{P}_k)(\mathbf{c}^k \mathbf{P}_k)^T = \left\| \mathbf{c}^k \mathbf{P}_k \right\|^2 \geq 0 ,
\end{aligned}$$

where $\left\| \mathbf{c}^k \mathbf{P}_k \right\|$ is the length of the row vector $\mathbf{c}^k \mathbf{P}_k$.

Having thus shown that $\mathbf{d}^k := \mathbf{P}_k (\mathbf{c}^k)^T$ is an improving direction, we now have to determine the step length α that guarantees that the next solution is still feasible. In order to preserve nonnegativity it is necessary that $\tilde{\mathbf{x}}^k + \alpha \mathbf{d}^k \geq \mathbf{0}$. As $\tilde{\mathbf{x}}^k = \mathbf{e}$, we must have $1+\alpha d_j^k \geq 0$, or, equivalently, $\alpha d_j^k \geq -1 \ \forall\, j$. For any nonnegative component d_j^k this is trivially true, and we therefore consider only elements $d_j^k < 0$, for which the above condition reduces to $\alpha \leq -\dfrac{1}{d_j^k}$ for all negative d_j^k. (In case $d_j^k \geq 0 \ \forall\, j$, we realize that the problem has unbounded "optimal" solutions). The largest possible value of α, here denoted by $\alpha_{\max}$, will therefore be

$$\alpha_{\max} = \min_j \left\{ -\frac{1}{d_j^k} : d_j^k < 0, j = 1,...,n \right\}.$$

Since the next point should remain in the interior of the feasible set, we may decide to move only some fraction $\beta < 1$ of the full distance to the boundary, i.e., $\alpha_k := \beta \alpha_{\max}$, so that the next interior point $\tilde{\mathbf{x}}^{k+1}$ is determined as

$$\tilde{\mathbf{x}}^{k+1} := \tilde{\mathbf{x}}^k + \alpha_k \mathbf{d}^k .$$

For instance, we may choose $\beta = 0.99$. In terms of the original variables and before the scaling operation, we obtain

$$\mathbf{x}^{k+1} = \mathbf{T}_k \tilde{\mathbf{x}}^{k+1} = \mathbf{T}_k \left(\tilde{\mathbf{x}}^k + \alpha_k \mathbf{d}^k \right) = \mathbf{x}^k + \alpha_k \mathbf{T}_k \mathbf{d}^k ,$$

and the next iteration can commence.

Before we summarize the above discussion in algorithmic form, we will consider how the calculations involving the projection matrix $\mathbf{P}_k$ can conveniently be handled. Setting $\mathbf{r}^k := \left(\mathbf{A}_k \mathbf{A}_k^T \right)^{-1} \mathbf{A}_k (\mathbf{c}^k)^T$, we obtain for the feasible direction

$$\begin{aligned} \mathbf{d}^k &= \mathbf{P}_k (\mathbf{c}^k)^T = \left(\mathbf{I} - \mathbf{A}_k^T \left(\mathbf{A}_k \mathbf{A}_k^T \right)^{-1} \mathbf{A}_k \right) (\mathbf{c}^k)^T = \\ &= (\mathbf{c}^k)^T - \mathbf{A}_k^T \left(\mathbf{A}_k \mathbf{A}_k^T \right)^{-1} \mathbf{A}_k (\mathbf{c}^k)^T = \\ &= (\mathbf{c}^k)^T - \mathbf{A}_k^T \mathbf{r}^k = \mathbf{T}_k \left(\mathbf{c}^T - \mathbf{A}^T \mathbf{r}^k \right), \end{aligned}$$

where $\mathbf{r}^k = \left(\mathbf{A}_k \mathbf{A}_k^T \right)^{-1} \mathbf{A}_k (\mathbf{c}^k)^T$, i.e., $\mathbf{A}_k \mathbf{A}_k^T \mathbf{r}^k = \mathbf{A}_k (\mathbf{c}^k)^T$ or $\mathbf{A} \mathbf{T}_k^2 \mathbf{A}^T \mathbf{r}^k = \mathbf{A} \mathbf{T}_k^2 \mathbf{c}^T$; this is the equation for solving $\mathbf{r}^k$. Therefore, we need not compute the matrix $\mathbf{P}_k$ itself.

We are now ready to formally state the interior point affine scaling method in algorithmic form. We assume that in a Phase 1 version of the method (to be described below), an interior point $\mathbf{x}^1 > \mathbf{0}$ has been found, so that Phase 2 can now commence. Recall that the problem is

$$\begin{aligned} \text{P: Max } z &= \mathbf{cx} \\ \text{s.t.} \quad & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}, \end{aligned}$$

where the constraint matrix $\mathbf{A}$ is assumed to have full rank. The algorithm is initialized with $\mathbf{x}^1 > \mathbf{0}$, such that $\mathbf{Ax}^1 = \mathbf{b}$ and the iteration counter is set to $k := 1$. The constant β in Step 4 below is preselected as a value less than one, e.g., $\beta = 0.99$.

Interior Point Affine Scaling Method: Phase 2

Step 1 (Scaling): Define the scaling matrix

$$\mathbf{T}_k = \text{diag } (\mathbf{x}^k) = \begin{bmatrix} x_1^k & 0 & \dots & 0 \\ 0 & x_2^k & \dots & 0 \\ \vdots & \dots & \dots & \vdots \\ 0 & 0 & 0 & x_n^k \end{bmatrix}.$$

Step 2: Find the solution $\mathbf{r}^k$ to the system of equations

$$\mathbf{A}\mathbf{T}_k^2\mathbf{A}^T\mathbf{r}^k = \mathbf{A}\mathbf{T}_k^2\mathbf{c}^T$$

and determine the improving feasible direction $\mathbf{d}^k$ as

$$\mathbf{d}^k := \mathbf{T}_k\left(\mathbf{c}^T - \mathbf{A}^T\mathbf{r}^k\right).$$

Step 3: Is $\mathbf{d}^k \geq \mathbf{0}$?
If yes: Stop, unbounded "optimal" solutions exist.
If no: Go to Step 4.

Step 4: Determine the step length

$$\alpha_k = \beta \min_j \left\{ -\frac{1}{d_j^k} : d_j^k < 0, j = 1,\ldots,n \right\}$$

and compute the next interior point as $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k\mathbf{T}_k\mathbf{d}^k$.

Step 5: Is some stop criterion satisfied?
If yes: Stop with the solution $\mathbf{x}^{k+1}$.
If no: Set $k := k+1$ and go to Step 1.

In the above algorithm, the stop criterion could simply be the number of iterations that are carried out. Another possibility is to stop, if $\left\|\mathbf{x}^{k+1} - \mathbf{x}^k\right\| < \varepsilon_1$ for some sufficiently small value $\varepsilon_1 > 0$. Yet another possibility is to terminate the algorithm if $z^{k+1} - z^k < \varepsilon_2$, i.e., if the improvement of the objective function falls short of some preset value $\varepsilon_2 > 0$. Since the algorithm terminates with an interior point, the task is now to determine a "nearby" extreme point. There are various ways to achieve this goal. One obvious way is to set up a simplex tableau for the problem and pivot in the variables in decreasing order of magnitude in the solution $\mathbf{x}^{k+1}$ that was obtained at the exit of the affine scaling method. Alternatively, we may use some method for finding the nearest extreme point and pivot as required. For details, see Dantzig and Thapa (2003). If the resulting point happens to be nonoptimal (which could happen in case the stop criterion caused the affine scaling method to stop prematurely), one or more pivot steps with the simplex method might be required. Yet another method would be to use $\mathbf{x}^{k+1}$ as a direction for the external column in external pivoting, see the discussion in Section 7.1 of this volume.

Assume now that an initial interior point $\mathbf{x}^1$ is not readily available. In such a case, some Phase 1 procedure of the affine scaling method is required. The idea for such a method is to consider the following problem with an initial point $\mathbf{x}^0 > \mathbf{0}$, which could, e.g., be the summation column vector $\mathbf{e} = [1, 1, \ldots, 1]^T$. Denote by A an artificial variable and note that unlike the Phase 1 method of the simplex

method, we will never need more than a single artificial variable in this method. If the positive initial point $\mathbf{x}^0$ satisfies the constraints, i.e., if $\mathbf{A}\mathbf{x}^0 = \mathbf{b}$, we can directly move to Phase 2 with $\mathbf{x}^1 := \mathbf{x}^0$. If this is not the case, then $\mathbf{b} - \mathbf{A}\mathbf{x}^0 \neq \mathbf{0}$ and we consider the problem

$$
\begin{aligned}
P_1: \ \min_{A,\mathbf{x}} \ & A \\
\text{s.t. } & \mathbf{A}\mathbf{x} + A(\mathbf{b} - \mathbf{A}\mathbf{x}^0) = \mathbf{b} \\
& \mathbf{x} \geq \mathbf{0} \\
& A \geq 0.
\end{aligned}
$$

It is easy to verify that $\mathbf{x} = \mathbf{x}^0$, $A = 1$ is an interior and feasible solution to problem P_1. However, as by assumption $\mathbf{A}\mathbf{x}^0 \neq \mathbf{b}$, we obtain $\mathbf{A}\mathbf{x} \neq \mathbf{b}$ as well which violates feasibility of the original problem. On the other hand, if $A \approx 0$, then $\mathbf{A}\mathbf{x} = \mathbf{b}$ is required, which indicates that $\mathbf{x}$ is indeed a feasible solution. This shows the validity of the above approach. Note that since we only deal with interior points, $\mathbf{x} > 0$ and $A > 0$ will always hold. We can then formally describe Phase 1 of the method. The method is initialized with a point $\mathbf{x} > \mathbf{0}$, e.g., $\mathbf{x}^0 = \mathbf{e}$.

Interior Point Affine Scaling Method: Phase 1

Step 1: Is $\mathbf{A}\mathbf{x}^0 = \mathbf{b}$?
If yes: Stop, set $\mathbf{x}^1 := \mathbf{x}^0$ and commence with Phase 2.
If no: Go to Step 2.

Step 2: Using the Phase 2 version of the affine scaling method, solve the problem

$$
\begin{aligned}
P_1: \ \min_{A,\mathbf{x}} \ & A \\
\text{s.t. } & \mathbf{A}\mathbf{x} + A(\mathbf{b} - \mathbf{A}\mathbf{x}^0) = \mathbf{b} \\
& \mathbf{x} \geq \mathbf{0} \\
& A \geq 0.
\end{aligned}
$$

As an initial feasible solution we can use $\mathbf{x} := \mathbf{x}^0$, $A := 1$. Denote an optimal solution to the problem P_1 by $(\mathbf{x}^1, \overline{A})$.

Step 3: Is $\overline{A} > 0$?
If yes: Stop, the problem P has no feasible solution.
If no: The point $\mathbf{x}^1$ is an interior feasible point that can be used as a feasible starting point in Phase 2.

In order to illustrate the above method, we will employ the same example that was used to demonstrate the two-phase simplex method in Chapter 3 of this volume. For convenience, we restate the problem here.

Example: Consider the following linear programming problem

$$\text{P: Max } z = 3x_1 + x_2$$

$$\begin{aligned} \text{s.t.} \quad & 3x_1 + 2x_2 \le 24 \\ & 4x_1 - x_2 \ge 8 \\ & x_1 - 2x_2 = 0 \\ & x_1, \quad x_2 \ge 0. \end{aligned}$$

In order to initialize Phase 1 of the Interior Point Affine Scaling Method, we first have to transform all structural equations into equations by adding a slack variable x_3 to the left-hand side of the first constraint and subtracting an excess variable x_4 from the left-hand side of the second equation (artificial variables are not needed here). The constraints are then

$$\begin{aligned} & 3x_1 + 2x_2 + x_3 = 24 \\ & 4x_1 - x_2 - x_4 = 8 \\ & x_1 - 2x_2 = 0, \end{aligned}$$

so that

$$\mathbf{A} = \begin{bmatrix} 3 & 2 & 1 & 0 \\ 4 & -1 & 0 & -1 \\ 1 & -2 & 0 & 0 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 24 \\ 8 \\ 0 \end{bmatrix},$$

and the objective function gradient vector $\mathbf{c} = [3, 1, 0, 0]$. Since no initial strictly positive solution $\mathbf{x}$ is readily available, we will start with Phase 1 of the affine scaling method, setting $\mathbf{x}^0 = \mathbf{e} = [1, 1, 1, 1]^T > \mathbf{0}$. The algorithm then proceeds as follows. In Step 1, we find that $\mathbf{A}\mathbf{x}^0 = [6, 2, -1]^T \neq [24, 8, 0]^T = \mathbf{b}$, and we go to Step 2.

Here, we maximize $-A$ instead of minimizing A and obtain the problem

$$\text{P}_1\text{: Max } -A$$

$$\text{s.t.} \begin{bmatrix} 3 & 2 & 1 & 0 \\ 4 & -1 & 0 & -1 \\ 1 & -2 & 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 18 \\ 6 \\ 1 \end{bmatrix} A = \begin{bmatrix} 24 \\ 8 \\ 0 \end{bmatrix}$$

$$x_j \ge 0, j=1, \ldots, 4; A \ge 0.$$

Writing this system in augmented form, we have an initial positive feasible

solution $\mathbf{x}^0 = [1, 1, 1, 1, 1]^T$, $\mathbf{c} = [0, 0, 0, 0, -1]$ and $\mathbf{A} = \begin{bmatrix} 3 & 2 & 1 & 0 & 18 \\ 4 & -1 & 0 & -1 & 6 \\ 1 & -2 & 0 & 0 & 1 \end{bmatrix}$.

Now $\mathbf{T}_0 = \text{diag}(\mathbf{x}^0) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$, so that $\mathbf{T}_0^2 = \mathbf{T} = \mathbf{I}$, and

$$\mathbf{A}\mathbf{T}_0^2\mathbf{A}^T = \begin{bmatrix} 338 & 118 & 17 \\ 118 & 54 & 12 \\ 17 & 12 & 6 \end{bmatrix}, \quad \mathbf{A}\mathbf{T}_0^2\mathbf{c}^T = \begin{bmatrix} -18 \\ -6 \\ -1 \end{bmatrix}. \text{ With}$$

$$(\mathbf{A}\mathbf{T}_0^2\mathbf{A}^T)^{-1} = \begin{bmatrix} 0.018304 & -0.051251 & 0.050641 \\ -0.051251 & 0.176835 & -0.208460 \\ 0.050641 & -0.208460 & 0.440106 \end{bmatrix}, \text{ we then find}$$

$$\mathbf{r}^0 = (\mathbf{A}\mathbf{T}_0^2\mathbf{A}^T)^{-1}\mathbf{A}\mathbf{T}_0^2\mathbf{c}^T = \begin{bmatrix} -0.0726052 \\ 0.0699614 \\ -0.1008745 \end{bmatrix}, \text{ and then}$$

$$\mathbf{d}^0 = \mathbf{T}_0(\mathbf{c}^T - \mathbf{A}^T\mathbf{r}^0) = \begin{bmatrix} 0.0388448 \\ 0.0134228 \\ 0.0726052 \\ 0.0699614 \\ -0.0119992 \end{bmatrix}.$$

The only negative element of this search direction is d_5^0 = -0.0119992, so that with $\beta = 0.99$, the step length is $\alpha_0 = 0.99/0.0119992 = 82.5056$.

We then find $\mathbf{x}^1 = \mathbf{x}^0 + \alpha_0\mathbf{T}_0\mathbf{d}^0 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} + 82.5056\mathbf{d}^0 = \begin{bmatrix} 4.2049 \\ 2.1075 \\ 6.9903 \\ 6.7722 \\ 0.01 \end{bmatrix}$.

With the artificial variable $A = 0.01$, we consider feasibility to be achieved, since we have deliberately avoided reaching zero. Rounding off, for ease of exposition, we find $x_1^1 = 4, x_2^1 = 2,$ and therefore $x_3^1 = 8$ and $x_4^1 = 6$. Summarizing, $\mathbf{x}^1$ = [4, 2, 8, 6]^T > 0, and Phase 1 is completed.

Phase 2 is then initialized with $\mathbf{x}^1$. In Step 1, we have $\mathbf{T}_1 = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 6 \end{bmatrix}$, and

with $\mathbf{A} = \begin{bmatrix} 3 & 2 & 1 & 0 \\ 4 & -1 & 0 & -1 \\ 1 & -2 & 0 & 0 \end{bmatrix}$ and $\mathbf{c}$ = [3, 1, 0, 0], we obtain

$$\mathbf{A}\mathbf{T}_1^2\mathbf{A}^T = \begin{bmatrix} 224 & 184 & 32 \\ 184 & 296 & 72 \\ 32 & 72 & 32 \end{bmatrix} \text{ and } \mathbf{A}\mathbf{T}_1^2\mathbf{c}^T = \begin{bmatrix} 152 \\ 188 \\ 40 \end{bmatrix}. \text{ With}$$

$$(\mathbf{A}\mathbf{T}_1^2\mathbf{A}^T)^{-1} = \begin{bmatrix} 0.01016383 & -0.00849515 & 0.00895024 \\ -0.00849515 & 0.01456311 & -0.02427184 \\ 0.00895024 & -0.02427184 & 0.07691141 \end{bmatrix},$$

we then find in Step 2

$$\mathbf{r}^1 = (\mathbf{A}\mathbf{T}_1^2\mathbf{A}^T)^{-1}\mathbf{A}\mathbf{T}_1^2\mathbf{c}^T = \begin{bmatrix} 0.3058252 \\ 0.4757282 \\ -0.1262136 \end{bmatrix}. \text{ Next,}$$

$$\mathbf{d}^1 = \mathbf{T}_1(\mathbf{c}^T - \mathbf{A}^T\mathbf{r}^1) = \mathbf{T}_1 \begin{bmatrix} 0.3058252 \\ 0.6116505 \\ -0.3058252 \\ 0.4757282 \end{bmatrix} = \begin{bmatrix} 1.223301 \\ 1.223301 \\ -2.446602 \\ 2.854369 \end{bmatrix}.$$

Here, only $d_3^1 = -2.446602$ is negative, so that with $\beta = 0.99$ the step length in Step 4 is $\alpha_1 = 0.99/2.446602 = 0.404643$, and

$$\mathbf{x}^2 = \mathbf{x}^1 + \alpha_1 \mathbf{T}_1 \mathbf{d}^1 = \begin{bmatrix} 4 \\ 2 \\ 8 \\ 6 \end{bmatrix} + 0.404643 \begin{bmatrix} 4.893204 \\ 2.446602 \\ -19.57282 \\ 17.12622 \end{bmatrix} = \begin{bmatrix} 5.98 \\ 2.99 \\ 0.08 \\ 12.93 \end{bmatrix}.$$

Here, we see that $x_1^2 = 5.98$ and $x_2^2 = 2.99$ and we terminate the procedure. It so happens that we are already quite close to the true optimal point $\bar{x}_1 = 6, \bar{x}_2 = 3$.

The progress of the method is shown in Figure 7.5.

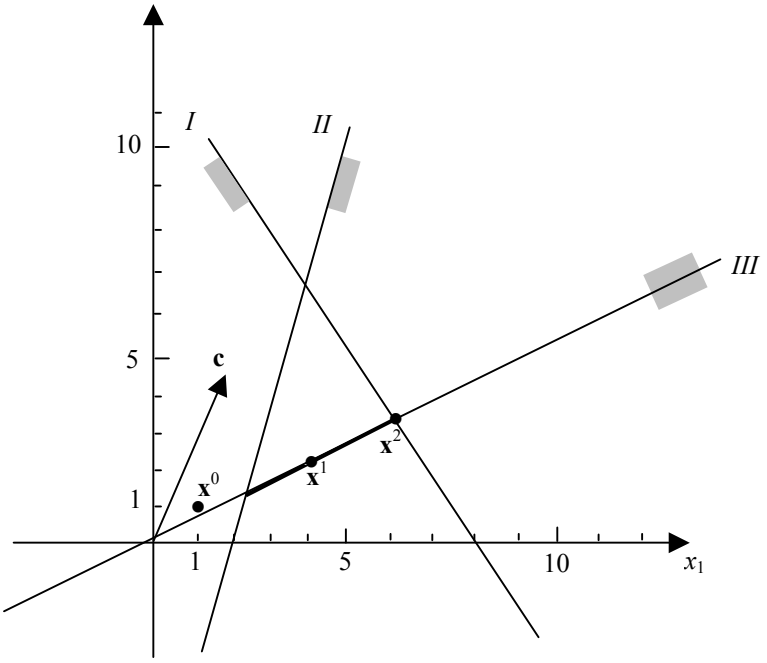


Figure 7.5

To demonstrate the effects of scaling graphically, we transform the point (x_1^1, x_2^1) = (4, 2) into the point $(\tilde{x}_1^1, \tilde{x}_2^1)$ = (1, 1), and show how the constraints *I*, *II*, and *III* appear in the scaled $(\tilde{x}_1, \tilde{x}_2)$-space. The problem is then

$$\tilde{P}: \text{ Max } \tilde{z} = 12\tilde{x}_1 + 2\tilde{x}_2$$

$$\text{s.t.} \qquad 12\tilde{x}_1 + 4\tilde{x}_2 \le 24 \qquad I'$$

$$16\tilde{x}_1 - 2\tilde{x}_2 \geq 8 \qquad II''$$
$$4\tilde{x}_1 - 4\tilde{x}_2 = 0 \qquad III''$$
$$\tilde{x}_1, \quad \tilde{x}_2 \geq 0.$$

which can be visualized in Figure 7.6.

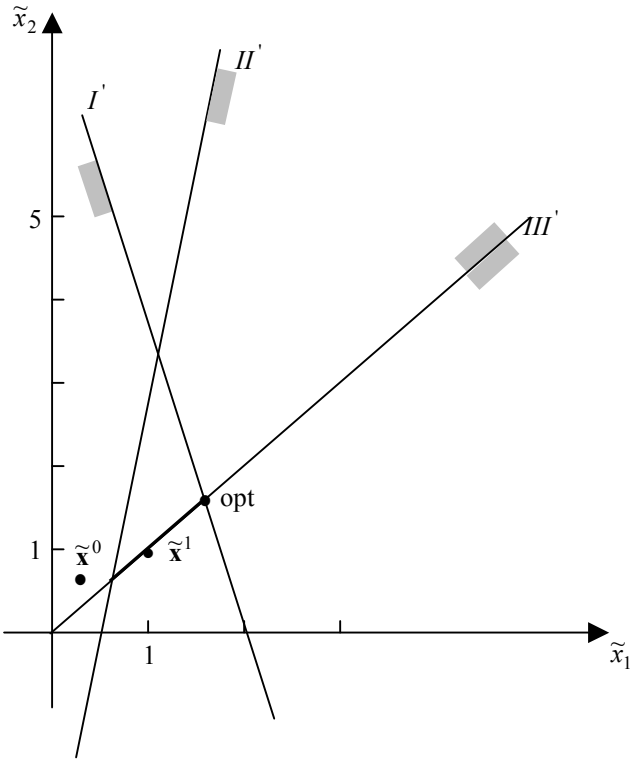


Figure 7.6

Although the affine scaling method in the above version does not have a polynomial worst-case bound, the similar method of Karmarkar (1984) does. With that method, versions have been described with a complexity of $O(n^{3.5}L)$, where L is the length of a binary encoding of the problem. For details, readers are referred to Roos *et al*. (2006).

The affine scaling method described above moves towards an optimal point in steps, always making sure that solutions remain in the relative interior of the feasible region. This property was guaranteed by deliberately stopping short of the boundary, even though we know by way of Dantzig's corner point theorem (see Lemma 3.1) that at least one optimal solution will be located at an extreme point (and it is certain that no optimal solution will be in the interior of the feasible set).

Another way to ensure that we stay away from the boundary of the feasible set is to use a *barrier method*, which introduces boundary repulsion terms. For an instructive survey regarding barrier methods, see, e.g., Fiacco (1979). Here, we will use the logarithmic barrier method for the purpose. More specifically, we add logarithmic barrier terms $\ln x_j$ to the objective function $\mathbf{cx} = \sum_{j=1}^{n} c_j x_j$, so that we obtain $z = \sum_{j=1}^{n} c_j x_j + \mu \sum_{j=1}^{n} \ln x_j$, where μ is some prespecified positive constant. Since $\ln x_j$ approaches $-\infty$ as x_j approaches 0 and since it is assumed that the objective function is of the maximization type, the effect of the barrier term is to force an optimal solution away from $x_j = 0$ for any variable x_j. Fiacco and McCormick (1968) have shown that the resulting maximal point will depend on the choice of μ, and if μ were to be allowed to parametrically approach zero, the optimal point would tend to the true optimum for the original objective function $z = \mathbf{cx}$. The problem, formulated with the barrier term is then

$$\begin{aligned} P^{\text{Barrier}}\text{: Max } z &= \mathbf{cx} + \mu \sum_{j=1}^{n} \ln x_j \\ \text{s.t.} \quad & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}, \end{aligned}$$

where the nonnegativity constraints $\mathbf{x} \geq 0$ can now be dropped, since the barrier terms will force x to have strictly positive components. As its objective function is concave and the constraints are linear, the problem P^{Barrier} is a concave maximization problem. Results from the theory of nonlinear optimization (see, e.g., Eiselt *et al*., 1987 or Bazaraa *et al*. (1993) assert that necessary and sufficient conditions for the optimality of $\bar{\mathbf{x}}$ for P^{Barrier} is the existence of a multiplier row vector $\bar{\mathbf{u}} \in \mathbb{R}^m$ such that

$$\mathbf{A}\bar{\mathbf{x}} = \mathbf{b} \tag{1}$$

$$(\bar{\mathbf{u}}\mathbf{A})_j - \mu / \bar{x}_j = c_j \tag{2}$$

$$\bar{\mathbf{x}} > \mathbf{0}\,. \tag{3}$$

Defining $\bar{s}_j = \mu / \bar{x}_j \ \forall j$, we can then write the problem (1), (2), and (3) as

$$\mathbf{A}\bar{\mathbf{x}} = \mathbf{b},\ \bar{\mathbf{x}} > \mathbf{0}$$

$$\bar{\mathbf{u}}\mathbf{A} - \bar{\mathbf{s}} = \mathbf{c},\ \bar{\mathbf{s}} > \mathbf{0}$$

$$\bar{s}_j \bar{x}_j = \mu \ \forall j.$$

Since μ is a preselected positive number, we can drop the conditions $\bar{\mathbf{x}} > \mathbf{0}$ and $\bar{\mathbf{s}} > \mathbf{0}$. For notational ease, we drop the bar over $\bar{\mathbf{x}}$ and $\bar{\mathbf{s}}$. Furthermore, we

introduce the diagonal matrices **X** and **S** which are defined as

$$\mathbf{X} := \begin{bmatrix} x_1 & 0 & \cdots & 0 \\ 0 & x_2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & x_n \end{bmatrix} \text{ and } \mathbf{S} := \begin{bmatrix} s_1 & 0 & 0 & 0 \\ 0 & s_2 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & s_n \end{bmatrix}.$$

The optimality conditions (1), (2), and (3) can then be written as the system

$$\mathbf{Ax} = \mathbf{b} \tag{4}$$
$$\mathbf{uA} - \mathbf{s} = \mathbf{c} \tag{5}$$
$$\mathbf{SXe} = \mu\mathbf{e} \tag{6}$$

It is interesting to note that the original problem

$$\begin{aligned} \text{P: Max } z &= \mathbf{cx} \\ \text{s.t.} \quad & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned}$$

has the dual

$$\begin{aligned} \text{P}_\text{D}\text{: Min } z_\text{D} &= \mathbf{ub} \\ \text{s.t.} \quad & \mathbf{uA} \geq \mathbf{c} \\ & \mathbf{u} \in \mathbb{R}^m \end{aligned}$$

and the optimality conditions for the primal-dual pair (P, P_D) are

$$\mathbf{Ax} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0} \quad \text{(primal feasibility)} \tag{7}$$
$$\mathbf{uA} \geq \mathbf{c},\ \mathbf{u} \in \mathbb{R}^m \quad \text{(dual feasibility)} \tag{8}$$
$$(\mathbf{uA}-\mathbf{c})\mathbf{x} = 0 \quad \text{(complementary slackness)} \tag{9}$$

Introducing the vector of surplus variables $\mathbf{s} := \mathbf{uA} - \mathbf{c}$, we can see that the conditions (4), (5), and (6) and the optimality conditions (7), (8), and (9) are identical if $\mu = 0$, with the exception that $\mathbf{x} > \mathbf{0}$ in the former is replaced by $\mathbf{x} \geq \mathbf{0}$ in the latter. Intuitively, if $\mu > 0$ is decreasing to zero, the solution of (4), (5), and (6) should tend to that of the optimality conditions (7), (8), and (9). For any positive value of μ, the solution to (4), (5), and (6) will be interior in the sense that $\mathbf{x} > \mathbf{0}$ and $\mathbf{s} > \mathbf{0}$, and the idea is to solve the system (4), (5), (6) of nonlinear equations with, e.g., the Newton-Raphson method and then successively let μ decrease to zero. We will actually reduce the value of μ at each step with the Newton-Raphson method.

Recall Procedure A.10 that describes the Newton-Raphson method for solving an equation $f(x) = 0$ by means of the iteration formula $x^{k+1} = x^k - f(x^k)/f'(x^k)$ for $k = 1$,

2, … . For the vector-valued version where we solve the system of nonlinear equations $f_i(\mathbf{x}) = 0$ for i=1, 2, …, m, with $\mathbf{x} \in \mathbb{R}^n$, we define $\mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{bmatrix}$ and the Jacobian matrix $\dfrac{\partial \mathbf{f}}{\partial \mathbf{x}} = \left[\dfrac{\partial f_i(\mathbf{x})}{\partial x_j} \right]$. Then the iteration formula is

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\left(\mathbf{x}^k\right) \right)^{-1} \mathbf{f}(\mathbf{x}^k), \quad k = 1, 2, \ldots,$$

which we will use in the form

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\left(\mathbf{x}^k\right)\left(\mathbf{x}^{k+1} - \mathbf{x}^k\right) = -\mathbf{f}(\mathbf{x}^k), \; k=1, 2, \ldots .$$

We begin by rewriting the optimality conditions (4), (5), and (6) as

$$\begin{array}{ll} \mathbf{Ax} - \mathbf{b} = \mathbf{0} & (m \text{ equations}) \\ \mathbf{A}^T\mathbf{u}^T - \mathbf{s}^T - \mathbf{c}^T = \mathbf{0} & (n \text{ equations}) \\ \mathbf{SXe} - \mu\mathbf{e} = \mathbf{0} & (n \text{ equations}) \end{array}$$

and we consider the (column) vector of variables to be $[\mathbf{x}^T, \mathbf{u}, \mathbf{s}]^T \in \mathbb{R}^{n+m+n}$. The Jacobian (first derivative) matrix for the left-hand side of this system becomes

$$\begin{bmatrix} \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}^T & -\mathbf{I} \\ \mathbf{S} & \mathbf{0} & \mathbf{X} \end{bmatrix}$$

with $m + n + n$ rows and $n + m + n$ columns, respectively, and the Newton iteration formulation is then

$$\begin{bmatrix} \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}^T & -\mathbf{I} \\ \mathbf{S}^k & \mathbf{0} & \mathbf{X}^k \end{bmatrix} \begin{bmatrix} \mathbf{x}^{k+1} - \mathbf{x}^k \\ (\mathbf{u}^{k+1})^T - (\mathbf{u}^k)^T \\ (\mathbf{s}^{k+1})^T - (\mathbf{s}^k)^T \end{bmatrix} = - \begin{bmatrix} \mathbf{A}\mathbf{x}^k - \mathbf{b} \\ \mathbf{A}^T(\mathbf{u}^k)^T - (\mathbf{s}^k)^T - \mathbf{c}^T \\ \mathbf{S}^k\mathbf{X}^k\mathbf{e} - \mu\mathbf{e} \end{bmatrix}.$$

Given the current solution $\mathbf{x}^k$, $\mathbf{u}^k$, $\mathbf{s}^k$, we will now solve the above system to obtain the new solution $\mathbf{x}^{k+1}$, $\mathbf{u}^{k+1}$, $\mathbf{s}^{k+1}$ and introduce the notation $\Delta\mathbf{x}^k = \mathbf{x}^{k+1} - \mathbf{x}^k$, $\Delta\mathbf{u}^k = \mathbf{u}^{k+1} - \mathbf{u}^k$, and $\Delta\mathbf{s}^k = \mathbf{s}^{k+1} - \mathbf{s}^k$. In order to keep the notation simple, we will write $\Delta\mathbf{x}$ instead of $\Delta\mathbf{x}^k$, and similarly $\Delta\mathbf{u}$, and $\Delta\mathbf{s}$. Furthermore, let $\Delta\mathbf{P} := \mathbf{A}\mathbf{x}^k - \mathbf{b}$ and $\Delta\mathbf{D}$

$:= \mathbf{A}^T(\mathbf{u}^k)^T - (\mathbf{s}^k)^T - \mathbf{e}^T$, again without superscript to keep the notation simple.

Then, with $\mathbf{S}$ instead of $\mathbf{S}^k$ and $\mathbf{X}$ instead of $\mathbf{X}^k$, Newton's formula can be written as

$$\mathbf{A}\Delta\mathbf{x} = -\Delta\mathbf{P} \tag{10}$$
$$\mathbf{A}^T\Delta\mathbf{u}^T - \Delta\mathbf{s}^T = -\Delta\mathbf{D} \tag{11}$$
$$\mathbf{S}\Delta\mathbf{x} + \mathbf{X}\Delta\mathbf{s}^T = \mu\mathbf{e} - \mathbf{SXe} \tag{12}$$

The task is now to solve the equations (10), (11), and (12) for $\Delta\mathbf{x}$, $\Delta\mathbf{u}$, and $\Delta\mathbf{s}$. In order to do so, premultiplying relation (11) by $\mathbf{X}$ and adding relation (12) yields $\mathbf{XA}^T\Delta\mathbf{u}^T + \mathbf{S}\Delta\mathbf{x} = -\mathbf{X}\Delta\mathbf{D} + \mu\mathbf{e} - \mathbf{SXe}$. Premultiplying this relation by $\mathbf{AS}^{-1}$ results in $\mathbf{AS}^{-1}\mathbf{XA}^T\Delta\mathbf{u}^T + \mathbf{A}\Delta\mathbf{x} = -\mathbf{AS}^{-1}\mathbf{X}\Delta\mathbf{D} + \mu\mathbf{AS}^{-1}\mathbf{e} - \mathbf{AXe}$. Relation (10) and the definition of $\Delta\mathbf{P}$ imply that $\mathbf{A}\Delta\mathbf{x} = -\Delta\mathbf{P} = \mathbf{b} - \mathbf{Ax}$, so that

$$\mathbf{AS}^{-1}\mathbf{XA}^T\Delta\mathbf{u}^T = -\mathbf{b} - \mathbf{AS}^{-1}\mathbf{X}\Delta\mathbf{D} + \mu\mathbf{AS}^{-1}\mathbf{e}$$

(note that $\mathbf{AXe} = \mathbf{Ax}$), from which $\Delta\mathbf{u}^T$ can be computed. From relation (11) we obtain

$$\Delta\mathbf{s}^T = \mathbf{A}^T\Delta\mathbf{u}^T + \Delta\mathbf{D},$$

and finally, by premultiplying relation (12) by $\mathbf{S}^{-1}$, we can solve for $\Delta\mathbf{x}$ via

$$\Delta\mathbf{x} = \mathbf{S}^{-1}(\mu\mathbf{e} - \mathbf{SXe} - \mathbf{X}\Delta\mathbf{s}^T) = \mathbf{S}^{-1}(\mu\mathbf{e} - \mathbf{X}\Delta\mathbf{s}^T) - \mathbf{x}.$$

Having now computed the Newton step directions $\Delta\mathbf{x}$, $\Delta\mathbf{u}$, and $\Delta\mathbf{s}$, we must ensure that $\mathbf{x} > \mathbf{0}$ and $\mathbf{s} > \mathbf{0}$. For $\mathbf{x}$, we use the step length α_P which must satisfy $x_j^k + \alpha_P\Delta x_j > 0$, or, equivalently, $\alpha_P\Delta x_j > -x_j^k \;\forall\, j$=1, …, n. Here, only $\Delta x_j < 0$ is of concern, so that we have to require that $\alpha_P < \min_j \left\{ \frac{-x_j^k}{\Delta x_j} : \Delta x_j < 0 \right\}$. In order to achieve the desired result, we set $\alpha_P = \beta \min_j \left\{ \frac{-x_j^k}{\Delta x_j} : \Delta x_j < 0 \right\}$ with some preselected $\beta < 1$. In this chapter, we typically use $\beta = 0.99$. Similarly, for the calculation of $\mathbf{s}$ we use the step length α_D determined by

$$\alpha_D = \beta \min_j \left\{ \frac{-s_j^k}{\Delta s_j} : \Delta s_j < 0 \right\}.$$

Here, we assume that $\Delta x_j < 0$ and $\Delta s_j < 0$ exist, avoiding difficulties with unboundedness and infeasibility. It is also possible to show that the duality gap

$\mathbf{u}^k\mathbf{b} - \mathbf{c}\mathbf{x}^k$ decreases in each Newton step; for details readers are referred to Marsten *et al*. (1990) or Roos *et al*. (2006) for an in-depth treatment.

It is also common practice to reduce the value of μ at each step of the algorithm. One way to accomplish this is to set the value of μ equal to some fraction of the current duality gap. For simplicity, we will follow Dantzig and Thapa (2003) and set $\mu := 10^{1-k}$; for a full discussion of more sophisticated updating of μ, see Roos *et al*. (2006).

We are now able to summarize the above discussion in algorithmic form. Recall that the problem under consideration is

$$\begin{aligned} \text{P: Max } z &= \mathbf{c}\mathbf{x} \\ \text{s.t.} \quad \mathbf{A}\mathbf{x} &= \mathbf{b} \\ \mathbf{x} &\geq \mathbf{0}. \end{aligned}$$

The associated dual problem is

$$\begin{aligned} \text{P}_\text{D}\text{: Min } z_D &= \mathbf{u}\mathbf{b} \\ \text{s.t.} \quad \mathbf{u}\mathbf{A} &\geq \mathbf{c} \\ \mathbf{u} &\in \mathbb{R}^m. \end{aligned}$$

For simplicity of the exposition, we assume here that an initial interior feasible solution is available that satisfies $\mathbf{x}^1 > \mathbf{0}$ and $\mathbf{A}\mathbf{x}^1 = \mathbf{b}$, and that an interior dual feasible solution $\mathbf{u}^1$ is also available with $\mathbf{u}^1\mathbf{A} - \mathbf{c} = \mathbf{s}^1 > \mathbf{0}$. Let $\mu := 1$ and set the iteration counter $k := 1$.

The Primal-Dual Newton Step Interior Point Method

Step 1: Is the scaled duality gap $\dfrac{\mathbf{u}^k\mathbf{b} - \mathbf{c}\mathbf{x}^k}{1 + |\mathbf{c}\mathbf{x}^1|} < \varepsilon$ for some preselected $\varepsilon > 0$?

If yes: Stop with the near-optimal solution $(\mathbf{x}^k, \mathbf{u}^k)$.
If no: Go to Step 2.

Step 2: Compute the Newton step direction $(\Delta\mathbf{x}, \Delta\mathbf{u}, \Delta\mathbf{s})$ as

$$\Delta\mathbf{u}^T = (\mathbf{A}\mathbf{S}^{-1}\mathbf{X}\mathbf{A}^T)^{-1}(-\mathbf{b} - \mathbf{A}\mathbf{S}^{-1}\mathbf{X}\Delta\mathbf{D} + \mu\mathbf{A}\mathbf{S}^{-1}\mathbf{e}), \text{ where}$$

$$\mathbf{X} = \begin{bmatrix} x_1^k & 0 & \cdots & 0 \\ 0 & x_2^k & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & x_n^k \end{bmatrix}, \quad \mathbf{S} = \begin{bmatrix} s_1^k & 0 & \cdots & 0 \\ 0 & s_2^k & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & s_n^k \end{bmatrix},$$

and $\Delta\mathbf{D} = \mathbf{A}^T(\mathbf{u}^k)^T - (\mathbf{s}^k)^T - \mathbf{c}^T$. Also, determine
$\Delta\mathbf{s}^T = \mathbf{A}^T\Delta\mathbf{u}^T + \Delta\mathbf{D}$ and
$\Delta\mathbf{x} = \mathbf{S}^{-1}(\mu\mathbf{e} - \mathbf{X}\Delta\mathbf{s}^T) - \mathbf{x}$.

Step 3: Compute the step lengths

$$\alpha_P = \beta \min_j \left\{ \frac{-x_j^k}{\Delta x_j} : \Delta x_j < 0 \right\} \text{ and}$$

$$\alpha_D = \beta \min_j \left\{ \frac{-s_j^k}{\Delta s_j} : \Delta s_j < 0 \right\}, \text{ the new point}$$

$x_j^{k+1} := x_j^k + \alpha_p \Delta\mathbf{x}$, j=1, 2, …, n,

$u_i^{k+1} := u_i^k + \alpha_D \Delta\mathbf{u}$, i=1, 2, …, m,

$s_j^{k+1} := s_j^k + \alpha_D \Delta\mathbf{s}$, j=1, 2, …, n, as well as the new barrier multiplier
$\mu := \mu/10$. Set $k := k + 1$ and go to Step 1.

We will now illustrate the primal-dual interior point method by solving the same example as in the affine scaling method above.

Example: Consider the linear programming problem

$$\begin{aligned} \text{P: Max } z &= 3x_1 + x_2 \\ \text{s.t.} \quad & 3x_1 + 2x_2 \le 24 \\ & 4x_1 - x_2 \ge 8 \\ & x_1 - 2x_2 = 0 \\ & x_1, \quad x_2 \ge 0. \end{aligned}$$

Adding slack variables x_3 and x_4 to the two constraints of the primal problem P, we obtain the linear programming problem in standard form

$$\begin{aligned} \text{P': Max } z' &= 3x_1 + x_2 \\ \text{s.t.} \quad & 3x_1 + 2x_2 + x_3 = 24 \\ & 4x_1 - x_2 - x_4 = 8 \\ & x_1 - 2x_2 = 0 \\ & x_j \ge 0, j = 1, \ldots, 4. \end{aligned}$$

Given the problem P', we can now determine its dual problem

$$\begin{aligned} P_D' : \text{ Min } z_D &= 24u_1 - 8u_2 \\ \text{s.t.} \quad & 3u_1 - 4u_2 + u_3 \ge 3 \\ & 2u_1 + u_2 - 2u_3 \ge 1 \\ & u_1 \ge 0 \end{aligned}$$

$$\begin{aligned} u_2 &\geq 0 \\ u_3 &\in \mathbb{R}. \end{aligned}$$

Subtracting excess variables $E_1^D, E_2^D, E_3^D,$ and E_4^D from all of the constraints of P_D' (including the two nonnegativity constraints), we obtain the dual problem in standard form as

$$\begin{aligned} P_D' : \text{Min } z_{D'} &= 24u_1 - 8u_2 \\ \text{s.t.} \quad 3u_1 - 4u_2 + u_3 - E_1^D &= 3 \\ 2u_1 + u_2 - 2u_3 - E_2^D &= 1 \\ u_1 - E_3^D &= 0 \\ u_2 - E_4^D &= 0 \\ u_i &\in \mathbb{R} \ \forall\ i = 1, \ldots, 4 \\ E_j^D &\geq 0 \ \forall\ j = 1, \ldots, 4. \end{aligned}$$

Here, we do not concern ourselves with the issue of finding an initial solution $\mathbf{x}^1$, $\mathbf{u}^1$, $\mathbf{E}^{D1}$ with $\mathbf{x}^1$, $\mathbf{E}^{D1} > 0$. As before, using the initial solution $\mathbf{x}^1 = [4, 2, 8, 6]^T > 0$, we find that setting $\mathbf{u}^1 = [2, ½, 0]$ results in $\mathbf{E}^{D1} = [1, 3½, 2, ½]$ as a feasible interior starting solution for P_D'. Set μ := 1 and β := 0.99, and we can calculate the scaled duality gap as

$$\frac{\mathbf{u}^1\mathbf{b} - \mathbf{c}\mathbf{x}^1}{1 + |\mathbf{c}\mathbf{x}^1|} = \frac{44 - 14}{1 + 14} = 2,$$

which is not small enough to stop, so we proceed to Step 2 of the algorithm. Now

$$\mathbf{c} = [3, 1, 0, 0], \ \mathbf{A} = \begin{bmatrix} 3 & 2 & 1 & 0 \\ 4 & -1 & 0 & -1 \\ 1 & -2 & 0 & 0 \end{bmatrix}, \ \mathbf{b} = \begin{bmatrix} 24 \\ 8 \\ 0 \end{bmatrix},$$

$$\text{as well as } \mathbf{X} = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 6 \end{bmatrix}, \text{ and } \mathbf{S} := \text{diag}(\mathbf{E}^D) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 3½ & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & ½ \end{bmatrix}.$$

We then find

$$\mathbf{AS}^{-1}\mathbf{XA}^T = \begin{bmatrix} 42\frac{2}{7} & 46\frac{6}{7} & 9\frac{5}{7} \\ 46\frac{6}{7} & 76\frac{4}{7} & 17\frac{1}{7} \\ 9\frac{5}{7} & 17\frac{1}{7} & 6\frac{2}{7} \end{bmatrix}, \text{ from which we can calculate}$$

$$[\mathbf{AS}^{-1}\mathbf{XA}^T]^{-1} = \begin{bmatrix} 0.0748175 & -0.0510949 & 0.0237226 \\ -0.0510949 & 0.0684307 & -0.1076642 \\ 0.0237226 & -0.1076642 & 0.4160584 \end{bmatrix}.$$

With dual feasibility $\Delta\mathbf{D} = 0$, so that

$$\Delta\mathbf{u}^T = [\mathbf{AS}^{-1}\mathbf{XA}^T]^{-1}[-\mathbf{b} + \mu\mathbf{AS}^{-1}\mathbf{e}] = \begin{bmatrix} -1.159672 \\ 0.541971 \\ 0.382299 \end{bmatrix},$$

$$\Delta\mathbf{s}^T = \mathbf{A}^T\Delta\mathbf{u}^T = \begin{bmatrix} -0.928832 \\ -3.625912 \\ -1.159672 \\ -0.541971 \end{bmatrix}, \text{ and}$$

$$\Delta\mathbf{x} = \mathbf{S}^{-1}(\mu\mathbf{e} - \mathbf{X}\Delta\mathbf{s}^T) - \mathbf{x} = \begin{bmatrix} 0.715328 \\ 0.357664 \\ -2.861314 \\ 2.503650 \end{bmatrix}.$$

Step 3 of the algorithm then determines $\alpha_P = 0.99\left(\frac{-8}{-2.861314}\right) = 2.767959$ and

$$\alpha_D = 0.99\ \min\left\{\frac{-1}{-0.928832}, \frac{-3½}{-3.625912}, \frac{-2}{-1.159672}, \frac{-½}{-0.541971}\right\} = 0.913333,$$

and finally the new solution

$$\mathbf{x}^2 = \mathbf{x}^1 + \alpha_P\Delta\mathbf{x} = \begin{bmatrix} 5.98 \\ 2.99 \\ 0.08 \\ 12.93 \end{bmatrix},$$

$$\mathbf{u}^2 = \mathbf{u}^1 + \alpha_D \Delta\mathbf{u} = [0.9408,\ -0.0050,\ 0.3492],$$

and

$$\mathbf{E}^{D2} = \mathbf{E}^{D1} + \alpha_D \Delta\mathbf{s} = [0.1516,\ 0.1882,\ 0.9408,\ 0.0050],$$

and then $\mathbf{x}^2$ is primal feasible and ($\mathbf{u}^2$, $\mathbf{E}^{D2}$) is dual feasible and the first iteration is now complete. It is now apparent that the primal solution $\mathbf{x}^2$ is the same that was obtained with the affine scaling method discussed earlier. The scaled duality gap is $\dfrac{\mathbf{u}^2\mathbf{b} - \mathbf{c}\mathbf{x}^2}{1 + |\mathbf{c}\mathbf{x}^2|} = \dfrac{22.5392 - 20.93}{1 + 20.93} = 0.0734$, much less than the duality gap of 2 that we achieved in the previous (initial) solution. At this point, we terminate the algorithm.

A few comments on computational aspects are in order. Although our discussion has presented the affine scaling and primal-dual interior point methods by using matrix inversion, this will be avoided in practice by solving the related set of simultaneous linear equations by using efficient numerical techniques known from linear algebra, most prominently the Choleski and Bunch Parlett (1971) factorizations of the matrices involved. For details, readers are referred to Dantzig and Thapa (2003) and Roos *et al*. (2006).

It appears that variants of the above primal-dual interior point method are the most successful of the interior point methods. Their computational complexity is about $O(n^3L)$, where L is the length of a binary encoding of the problem. It is reported that implementations of such methods rarely use more than 50 and mostly around 20 iterations for convergence to optimal solutions with 8-digit accuracy, see. e.g., Roos *et al*. (2006).

Finally some words regarding new developments of methods and software for linear (as well as integer and nonlinear) programming problems. An unfortunate side effect of the commercialization of algorithms has been the increasing tendency of research organizations to keep their findings on new methods and their implementation confidential, for competitive reasons. Whereas in the past new results were disseminated to the scientific community at large, today, methods that will enhance the performance of commercial software packages have a tendency to become proprietary and only gradually, if at all, becoming part of common scientific knowledge. Many statements regarding computational speed and accuracy will therefore have to be taken with a grain of salt if these statements are based on only partially disclosed information. It appears that the scientific community will have to live with this state of affairs for some time to come. A bright side of the story is that there is today a whole lot of research done that would not have been financially supported, were it not for profit-oriented organizations.

Appendix

Definition 7.2: A *projection matrix* $\mathbf{P}$ is a square matrix which is symmetric, i.e., $\mathbf{P}^T = \mathbf{P}$, and idempotent, i.e., $\mathbf{P}^2 = \mathbf{P}$.

Lemma 7.3: Let $\mathbf{P} = \mathbf{I} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A}$ be a matrix, where the matrix $\mathbf{A}$ is assumed to have full rank, so that $\mathbf{A}\mathbf{A}^T$ isnonsingular. Then $\mathbf{P}$ is a projection matrix.

Proof: Using elementary rules for operations on matrices (see Proposition A.7), we obtain

$$\mathbf{P}^T = \left(\mathbf{I} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A}\right)^T = \mathbf{I} - \left(\mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A}\right)^T =$$
$$\mathbf{I} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A} = \mathbf{P}\text{, so that } \mathbf{P} \text{ is symmetric. Furthermore,}$$
$$\mathbf{P}^2 = \mathbf{P}\mathbf{P} = \left(\mathbf{I} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A}\right)\left(\mathbf{I} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A}\right) =$$
$$\mathbf{I} - 2\mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A} + \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A}\mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A} =$$
$$\mathbf{I} - 2\mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A} + \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A} = \mathbf{I} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A} = \mathbf{P},$$

so that the matrix $\mathbf{P}$ is idempotent.□

8 PROBLEM REFORMULATIONS

This section investigates a number of scenarios that do not obviously fall into the realm of linear programming. These include specifications of variables, types of constraints, and objective functions that differ from the linear programming formulations in standard and canonical form as defined in Chapter 3.2. This chapter is divided into three sections. The first (short) section discusses *variables* that do not fit into the standard and canonical forms, the second section deals with *constraints*, and the third, and arguably most important, section considers a number of *objectives* that do not appear to lend themselves to linear programming.

8.1 Reformulations of Variables

This section will look at variables and how to deal with them if variables have simple bounds or are unrestricted.

8.1.1 Lower Bounding Constraints

First consider the case in which variables have simple upper or lower bounds attached with them. Such instances frequently occur in many practical settings, e.g., when refining the diet problem and in the process restricting the number of servings of individual foods that are considered "manageable." Clearly, lower bound constraints of the type $x_j \geq \ell_j$ and upper bound constraints $x_j \leq u_j$ can be written as regular constraints and included as such in a linear programming problem. On the other hand, there are more convenient ways to include such simple constraints in a formulation. First consider lower bounds on a variable x_j. We now have the lower bound constraints $x_j \geq \ell_j$ as well as the usual nonnegativity constraints $x_j \geq 0$. Defining a new variable $x'_j = x_j - \ell_j$, we can then replace the old variable x_j wherever it appears in the model by the new variable x'_j. The lower bound constraints $x_j \geq \ell_j$ can then be written as $x'_j + \ell_j \geq \ell_j$ or simply $x'_j \geq 0$, which are the new nonnegativity constraints that are considered implicitly without being written as a formal (or structural) constraint. This variable transformation

can be applied to all variables with lower bounds, thus limiting the size of the problem under consideration. Clearly, when the optimal solution of the model (with the new variables x'_j) has been obtained, we have to recalculate the values of the original variables x_j.

Unfortunately, there is no similar way to deal with upper bounding constraints. Instead, the simplex algorithm is slightly modified to consider the upper bounding constraints without explicitly including them in the model. A description of this technique is provided in Chapter 5.2 of this volume. A similar, but much more complex, modification is possible for generalized upper bounding constraints of the type $\sum x_j = 1$ with the summation taken over some subset of the set of variables. For details, reader are referred to Eiselt *et al*. (1987) and Cooper and Steinberg (1970).

8.1.2 Variables Unrestricted in Sign

Here, we deal with variables which are not restricted to be nonnegative, i.e., we have $x_j \in \mathbb{R}$ rather than $x_j \geq 0$. The simplest method to transform an unrestricted variable $x_j \in \mathbb{R}$ so that it fits into a problem in standard or canonical form is to decompose it into its positive part x_j^+ and its negative part x_j^-, respectively, and then to replace the original variable $x_j \in \mathbb{R}$ by $x_j := x_j^+ - x_j^-$ and $x_j^+, x_j^- \geq 0$. In other words, the variable x_j is replaced by the difference of x_j^+ and x_j^- wherever it appears in the model, in the objective function as well as in all constraints. The new model is then solved and then the optimal values of x_j^+ and x_j^- are used to reconstruct the optimal value of x_j. We also would like to point out that this procedure is used in goal programming (see Chapter 9.3.3), where x_j^+ and x_j^- have interesting interpretations.

The obvious drawback of this transformation is that it doubles the number of variables. An alternative is, however, available. First rearrange the variables, so that the first r variables are unrestricted in sign and the last $(n - r)$ variables are nonnegative, i.e., $x_j \in \mathbb{R}$, $j = 1, \ldots, r$ and $x_j \geq 0$, $j = r + 1, \ldots, n$. Now one new variable $y \geq 0$ as well as r new variables x'_j, $j = 1, \ldots, r$ are created so that $x_j := x'_j - y \; \forall \, j = 1, \ldots, r$. Then any given constraint $\sum_{j=1}^{n} a_{ij} x_j \; R_i \; b_i$ can be written as

$$\sum_{j=1}^{r} a_{ij}(x'_j - y) + \sum_{j=r+1}^{n} a_{ij} x_j \; R_i \; b_i$$

or, equivalently,

$$\sum_{j=1}^{r} a_{ij} x'_j + \sum_{j=r+1}^{n} a_{ij} x_j - y \sum_{j=1}^{r} a_{ij} \;\; R_i \;\; b_i \; .$$

A similar transformation is used in the objective function. The following example may illustrate the transformation.

Example: Consider the following linear programming problem:

$$\begin{array}{lrrrrrrrl}
\text{P: Min } z = & 5x_1 & - 4x_2 & + & x_3 & & & & \\
\text{s.t.} & -x_1 & + & x_2 & + & x_3 & \le & 12 \\
& x_1 & - 2x_2 & & + 2x_3 & & \ge & 4 \\
& & x_2 & & & & \ge & 2 \\
& x_1, & x_2 & & & & \in & \mathbb{R} \\
& & & & x_3 & & \ge & 0
\end{array}$$

Applying the naïve transformation, we obtain the equivalent problem

$$\begin{array}{lrrrrrrrrrrl}
\text{P': Min } z = & 5x_1^+ & - 5x_1^- & - 4x_2^+ & + 4x_2^- & + x_3 & & & & & & \\
\text{s.t.} & - x_1^+ & + x_1^- & + x_2^+ & - x_2^- & + x_3 & \le & 12 \\
& x_1^+ & - x_1^- & - 2x_2^+ & + 2x_2^- & + 2x_3 & \ge & 4 \\
& & & x_2^+ & - x_2^- & & \ge & 2 \\
& x_1^+, & x_1^-, & x_2^+, & x_2^-, & x_3 & \ge & 0,
\end{array}$$

which has the optimal solution $\bar{x}_1^+ = 0$, $\bar{x}_1^- = 4$, $\bar{x}_2^+ = 2$, $\bar{x}_2^- = 0$, and $\bar{x}_3 = 6$, so that $\bar{z} = -22$. The second transformation results in the problem

$$\begin{array}{lrrrrrrrl}
\text{P'': Min } z = & 5\,x'_1 & - 4x'_2 & + & x_3 & - y & & & \\
\text{s.t.} & -x'_1 & + & x'_2 & + & x_3 & \le & 12 \\
& x'_1 & - 2\,x'_2 & & + 2x_3 & + y & \ge & 4 \\
& x'_1, & & x'_2, & & x_3, \;\; y & \ge & 0,
\end{array}$$

which has an optimal solution $\bar{x}'_1 = 0$, $\bar{x}'_2 = 6$, $\bar{x}_3 = 6$ and $\bar{y} = 4$, so that again $\bar{z} = -22$. In both cases, we can reconstruct the solution to the original problem as $\bar{x}_1 = -4$, $\bar{x}_1 = 2$, and $\bar{x}_3 = 6$ with $\bar{z} = -22$.

The obvious advantage of the second method is that, no matter how many unrestricted variables exist, the problem size increases by only one variable.

Another possibility exists if the problem includes at least one equation and a variable that is unrestricted in sign and which has a nonzero coefficient in the column of the unrestricted variable. Let again $x_j \in \mathbb{R}$ and let the *i*-th constraint be an equality, so that $\sum_{k=1}^{n} a_{ik} x_k = b_i$. Given that $a_{ij} \neq 0$, that equality can be rewritten as $x_j = \frac{1}{a_{ij}} \left[b_i - \sum_{\substack{k=1 \\ k \neq j}}^{n} a_{ik} x_k \right]$ and x_j can be replaced by this expression wherever it appears in the model. This procedure may be repeated as often as necessary if more than one unrestricted variable exists. Note, however, that this transformation can only be used for unrestricted variables. Eliminating variables $x_j \geq 0$ without considering their nonnegativity constraints may lead to wrong results as shown in the following

Example: Let the following linear programming problem be given:

$$\begin{aligned} \text{P: Max } z &= 3x_1 + 5x_2 \\ \text{s.t.} \quad x_1 &+ 2x_2 \leq 10 \\ 2x_1 &+ 3x_2 = 6 \\ x_1, \quad x_2 &\geq 0 \end{aligned}$$

The optimal solution of the problem is $\bar{x}_1 = 0, \bar{x}_2 = 2$, with $\bar{z} = 10$. If we falsely use the equality to eliminate one of the variables, say $x_1 = 3 - \frac{3}{2} x_2$, we would obtain the single-variable problem

$$\begin{aligned} \text{P': Max } z' &= \tfrac{1}{2} x_2 + 9 \\ \text{s.t.} \quad \tfrac{1}{2} x_2 &\leq 7 \\ x_2 &\geq 0, \end{aligned}$$

which has the unique optimal solution $\bar{x}_2 = 14$ with $\bar{z}' = 16$. Reconstruction of the value of x_1 results in $\bar{x}_1 = 3 - \frac{3}{2} \bar{x}_2 = -18$, which clearly violates the nonnegative constraint for x_1, i.e., P' would be equivalent to P only if the constraint $x_1 \geq 0$ or, equivalently, $3 - \frac{3}{2} x_2 \geq 0$ or $\frac{3}{2} x_2 \leq 3$ were added to P', resulting in the optimal solution $\bar{x}_2 = 2$ and hence $\bar{x}_1 = 0$. If, however, the variable x_1 were unrestricted in sign, the above procedure would be correct.

8.2 Reformulations of Constraints

Similar to replacing a variable that is unrestricted in sign by two nonnegative variables, it is possible to convert an equation into two inequalities. In many ways,

our discussion here is dual to what was said above about variables. Formally consider an equation $\mathbf{a}_{i\bullet}\mathbf{x} = b_i$ which is represented by a hyperplane in $\mathbb{R}^n$. We can then replace this equality by the two inequalities $\mathbf{a}_{i\bullet}\mathbf{x} \leq b_i$ and $\mathbf{a}_{i\bullet}\mathbf{x} \geq b_i$, each defining a closed halfspace, so that their intersection is nothing but their dividing hyperplane. When such a conversion is applied, it is useful to include some leeway for rounding errors, i.e., replace the equality $\mathbf{a}_{i\bullet}\mathbf{x} = b_i$ by the inequalities $\mathbf{a}_{i\bullet}\mathbf{x} \leq b_i + \varepsilon$ and $\mathbf{a}_{i\bullet}\mathbf{x} \geq b_i - \varepsilon$ for a small $\varepsilon > 0$. Graphically speaking, we would not only consider points on the hyperplane but also points in a narrow "corridor" surrounding the hyperplane. Notwithstanding the validity of the above transformation, the serious drawback is that its application doubles the number of linear relations.

Analogous to the above discussion concerning variables, it is possible to transform a system of equalities regardless of its size into a equivalent system of inequalities by adding just a single linear relation. Suppose that the given (sub)-system is $\mathbf{Ax} = \mathbf{b}$, then an equivalent system is $\mathbf{Ax} \leq \mathbf{b}$, $\mathbf{eAx} \geq \mathbf{eb}$ where $\mathbf{e}$ is a summation vector. Again, we suggest to use $\mathbf{eAx} \geq \mathbf{eb} - \varepsilon$ with some small $\varepsilon > 0$ as the additional constraint. This transformation is especially suitable for the dual simplex method, see Chapter 5.1 of this volume. As an illustration, consider the following

Example: Consider the following linear programming problem:

$$\begin{aligned} \text{P: Max } z = \; & 3x_1 + 2x_2 + 5x_3 \\ \text{s.t.} \quad & 2x_1 - x_2 + 3x_3 = 6 \\ & -x_1 + 3x_2 - x_3 = 9 \\ & x_1, \quad x_2, \quad x_3 \geq 0. \end{aligned}$$

The problem can then be rewritten as the equivalent formulation

$$\begin{aligned} \text{P: Max } z = \; & 3x_1 + 2x_2 + 5x_3 \\ \text{s.t.} \quad & 2x_1 - x_2 + 3x_3 \leq 6 \\ & -x_1 + 3x_2 - x_3 \leq 9 \\ & x_1 + 2x_2 + 2x_3 \geq 15 \\ & x_1, \quad x_2, \quad x_3 \geq 0. \end{aligned}$$

Absolute values of the left–hand sides are dealt with in the remainder of this section. Formally, we consider the relation $|LHS|\ R_i\ RHS$. Whenever the right-hand side values are negative or zero, we either have no feasible solution, a redundant constraint, or the case in which the left-hand side must equal zero. None of these cases are of interest in practical implementations. Suppose now that a right-hand side value is strictly positive. If the relation is $R_i = \{=\}$, the feasible set defined by the constraint comprises all points on two parallel hyperplanes, i.e., a set that is obviously not convex and cannot be dealt with in the context of linear programming. The case $R_i = \{\geq\}$ is similar in that it describes two halfspaces whose bordering hyperplanes are parallel, so that the feasible set is the entire

space except for a corridor between two parallel hyperplanes. Again, this set is not convex and can thus not be modeled as a linear program. However, both cases can be dealt with by solving the resulting two linear programming problems separately and then choosing the solution with the better objective function value.

Let now $R_i = \{\leq\}$. The feasible set is again bounded by two parallel hyperplanes, except this time the feasible set includes all points between these two hyperplanes. The graph in Figure 8.1 shows the feasible set for the constraint $|x_1 + 2x_2| \leq 2$ as the shaded area.

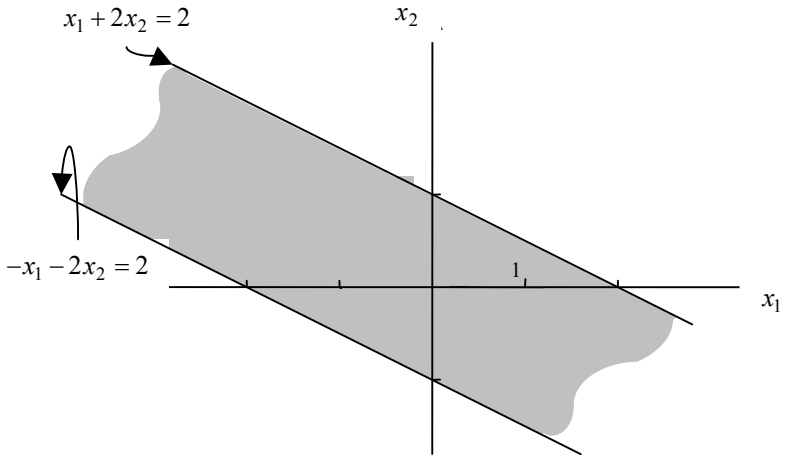


Figure 8.1

It is apparent that this set can be modeled by the two constraints $x_1 + 2x_2 \leq 2$ *and* $-x_1 - 2x_2 \leq 2$, i.e., in general, we can replace the constraint $|LHS| \leq RHS$ by the two linear constraints, $LHS \leq RHS$ and $-LHS \leq RHS$.

An extension is the case in which the left-hand side comprises the sum of absolute values, e.g., $\sum_j |\, a_{ij} - x_j \,| \leq b_i$ with $b_i > 0$. Such a relation can be replaced by the constraints $\sum_j (a_{ij} - x_j)(-1)^{\delta_j} \leq b_i \quad \forall \ \delta_j = 0 \vee 1$, which is an exponential number of constraints.

A special case of this reformulation occurs in the context of facility location problems, where a facility is to be located at an unknown point with coordinates x and y, and we attempt to measure the distance between the location of the facility

and the location of a known customer at some point with coordinates a and b. Define now the distance between the facility and the customer as the so-called *Manhattan distance*, which is defined as the sum of distances of both dimensions, i.e., $|x - a| + |y - b|$. Assuming that this distance should not exceed a prespecified limit z, we can write the constraint $|x - a| + |y - b| \leq z$, which is precisely the situation shown above with two terms on the left-hand side. As each of the two absolute values can be positive or negative, we will have to replace this constraint by four linear constraints:

$$\begin{aligned} x - a + y - b &\leq z, \\ x - a - y + b &\leq z, \\ -x + a + y - b &\leq z, \text{ and} \\ -x + a - y + b &\leq z. \end{aligned}$$

8.3 Reformulations of the Objective Function

This section deals with a variety of objective functions that, at least in their original form, are not part of a linear programming problem. However, as we will demonstrate, they can be reformulated so that they do fit the mold of standard linear programming.

8.3.1 Minimize the Weighted Sum of Absolute Values

Optimization of the sum of weighted absolute values can only be performed under certain conditions with the help of linear programming. The problem can be stated as

$$\begin{aligned} \text{P: Min } z &= \sum_{j=1}^{n} c_j \left| x_j \right| \\ \text{s.t.} \quad & \mathbf{Ax} \leq \mathbf{b} \\ & \mathbf{x} \in \mathbb{R}^n. \end{aligned}$$

Consider now a contour line of the objective for some given value of the objective function. Let this value be $\bar{z}$ and suppose the objective function is written as $z(x)$. The contour line in the $(x_1, x_2, \ldots, x_n)$ space of the decision variables is then a hyperdiamond. The set $z(x) \leq \bar{z}$ is then convex, while the set $z(x) \geq \bar{z}$ is not. We then write

Definition 8.1: A function is *convex* if for any $\mathbf{x}^1 + \mathbf{x}^2$ and for any real number $\lambda \in]0, 1[$ the relationship $f(\lambda\mathbf{x}^1 + (1-\lambda)\mathbf{x}^2) \leq \lambda f(\mathbf{x}^1) + (1-\lambda)f(\mathbf{x}^2)$ holds. The negative of a convex function is called concave.

From the definition of convexity we can prove

Lemma 8.2: For a convex function f, the set $\{\mathbf{x}: f(\mathbf{x}) \le \alpha\}$ is convex for any real number α. Similarly, for a concave function g, the set $\{\mathbf{x}: g(\mathbf{x}) \ge \beta\}$ is convex for any real number β.

It is apparent that the function $\sum_{j=1}^{n} c_j |x_j|$ is convex, as long as $c_j \ge 0 \ \forall j$, otherwise it is not. Similarly, the function $\sum_{j=1}^{n} c_j |x_j|$ is concave, as long as $c_j \le 0 \ \forall j$.

Example: Consider the function $z = |x_1| + 2|x_2|$.

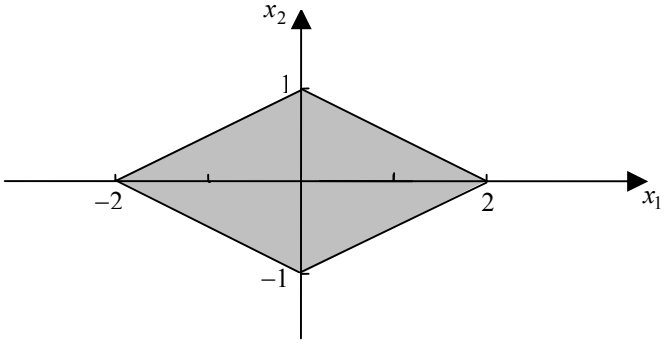


Figure 8.2

The shaded area in Figure 8.2 indicates the solutions (x_1, x_2) for which $z = |x_1| + 2|x_2| \le 2$. It is apparent that this is a convex set.

In general, it is easy to minimize a convex function (or, equivalently, maximize a concave function), whereas it is difficult to maximize a convex function (or minimize a concave function).

Assume now that we are minimizing a function $\sum_{j=1}^{n} c_j |x_j|$ with $c_j > 0 \ \forall j$. We can then proceed in a number of ways, two of which are outlined below.

Reformulation: Define new variables d_j^+ and d_j^- which denote the positive and negative value of the unrestricted variable x_j, respectively. Then $x_j = d_j^+ - d_j^-$ and $|x_j| = d_j^+ + d_j^-$, so that we can rewrite problem P above as

$$\text{P': Min } z' = \sum_{j=1}^{n} c_j (d_j^+ + d_j^-)$$

$$\text{s.t. } \mathbf{A}\mathbf{d}^+ - \mathbf{A}\mathbf{d}^- \leq \mathbf{b}$$

$$d_j^+ , \quad d_j^- \geq 0 \; \forall \, j.$$

This reformulation doubles the number of variables, while keeping the same number of constraints.

An application of the model discussed in this section is found in the reshipment model of transportation planning, see Section 2.8. Recall that the model was written as

$$\text{Min } z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} \mid x_{ij} \mid$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} = s_i \quad \forall \; i = 1,...,m$$

$$\sum_{i=1}^{m} x_{ij} = d_j \quad \forall \; j = 1,...,n$$

$$x_{ij} \in \mathbb{R} \quad \forall \; i=1, \ldots, m; j=1, \ldots, n,$$

where s_i denotes the supply at origin i, d_j is the demand at destination j, and the unit transportation costs c_{ij} naturally satisfy the required conditions $c_{ij} > 0$. The small example had two origins with supplies 10 and 10, respectively, as well as two destinations with demands of 5 and 15, respectively. Given the cost matrix

$$\mathbf{C} = \begin{bmatrix} 2 & 6 \\ 1 & 2 \end{bmatrix},$$

the reformulation is then

$$\text{P: Min } z' = 2\,d_{11}^+ + 2d_{11}^- + 6d_{12}^+ + 6d_{12}^- + d_{21}^+ + d_{21}^- + 2d_{22}^+ + 2d_{22}^-$$

$$\text{s.t.} \quad d_{11}^+ - d_{11}^- + d_{12}^+ - d_{12}^- = 10$$

$$d_{21}^+ - d_{21}^- + d_{22}^+ - d_{22}^- = 10$$

$$d_{11}^+ - d_{11}^- + d_{21}^+ - d_{21}^- = 5$$

$$d_{12}^+ - d_{12}^- + d_{22}^+ - d_{22}^- = 15$$

$$d_{11}^+, d_{11}^-, d_{12}^+, d_{12}^-, d_{21}^+, d_{21}^-, d_{22}^+, d_{22}^- \geq 0 \, .$$

Another application of the above transformation can be found in the area of curve fitting. Assume that we have obtained m observations with respect to n criteria. Each such observation is denoted by x_{ij}, $i = 1, \ldots, m$ and $j = 1, \ldots n$, which are referred to as independent variables. In addition we have corresponding observations y_i, $i = 1, \ldots, m$, which refer to our dependent variable. The idea is now to use the observations x_{ij} to estimate the dependent variable y. In order to do so, we first define points $P_i = (y_i; x_{i1}, x_{i2}, \ldots, x_{ij}, \ldots x_{in})$ as well as a hyperplane in (n+1)-dimensional space as $y = b + \sum_{j=1}^{n} a_j x_j$, where b and a_j, j=1, …, n are the n+1 parameters whose values are to be determined. The objective is to locate the hyperplane, so as to match the locations of the points P_i, i=1, …, m as closely as possible. The actual formulation of the objective will depend on the definition of proximity. One possibility is to minimize the sum of absolute deviations of the observed points from the hyperplane. This is the approach taken here. (Other possibilities are to measure the quality of the fit of the hyperplane as the sum of squared deviations or the maximal deviation of any of the observations from the hyperplane. For further details the reader is referred to Appa and Smith (1973) as well as Norback and Morris (1980), or the survey by Hobson and Weinkam (1979).

The deviation of the point P_i from the hyperplane (measured on the y axis) is denoted by $d_i = |y_i - y| = \left| y_i - b - \sum_{j=1}^{n} a_j x_{ij} \right|$. Minimizing the sum of these absolute values of the point – hyperplane deviations involves solving the problem

$$\begin{aligned} \text{P}: \text{Min } z &= \sum_{i=1}^{m} \left| y_i - b - \sum_{j=1}^{n} a_j x_{ij} \right| \\ \text{s.t.} \quad & b, a_j \in \mathbb{R}, j = 1, \ldots, n. \end{aligned}$$

Using the above transformation, we obtain the equivalent formulation

$$\begin{aligned} \text{P}': \text{min } z' &= \sum_{i=1}^{m} (d_i^+ + d_i^-) \\ \text{s.t.} \quad & d_i^+ - d_i^- + b + \sum_{j=1}^{n} a_j x_{ij} = y_i, \; i = 1, \ldots, m \\ & d_i^+, d_i^- \geq 0, i = 1, \ldots, m \\ & b, a_j \in \mathbb{R}, j = 1, \ldots, n. \end{aligned}$$

Some software packages require to express the variables b and a_j, j=1, …, n, in terms of nonnegative variables. This can be accomplished by using the transformation discussed in Section 8.1.2.

As an illustration, consider the following numerical

Example: It has been conjectured that the price of an upscale vehicle is a function of its gas mileage, horsepower, and weight. Table 8.1 provides the numerical information that has been taken from a number of internet sources.

Table 8.1

2007 model	MSRP price	Gas mileage (hwy), mpg	hp	weight (lbs)
Cadillac DTS	41,525	25	275	4,009
Corvette Z06, LZ1	69,175	26	505	3,132
Jaguar XJ Vanden Plas	74,835	27	300	3,819
Lexus GS 430	52,375	25	290	3,748

(where MSRP is the manufacturer's suggested retail price).

The formulation of the problem is then

$$\begin{aligned}\text{P: Min } z = |&(41{,}525 - b - 25a_1 - 275a_2 - 4{,}009a_3)\\ &+ (69{,}175 - b - 26a_1 - 505a_2 - 3{,}132a_3)\\ &+ (74{,}835 - b - 27a_1 - 300a_2 - 3{,}819a_3)\\ &+ (52{,}375 - b - 25a_1 - 290a_2 - 3{,}748a_3)|\end{aligned}$$

$$\text{s.t. } b;\ a_1, a_2, a_3 \in \mathbb{R}.$$

Transforming the problem and solving the linear programming problem P' results in the regression hyperplane

$$(\text{price}) = -68{,}394.31 + 13{,}583.50\ (\text{mpg}) - 124(\text{hp}) - 48.7361\ (\text{lbs}),$$

where we have replaced the original variables a_1, a_2, and a_3 by their optimal values that result from solving the problem P.

This somewhat surprising result indicates that each additional mile per gallon will cost \$13,583.50, while each additional horsepower saves \$124 (caused by the very powerful, but comparatively inexpensive Corvette) and each additional pound in weight saves \$48.74, a result that is again caused by the lightweight Corvette.

The ideas that led to the above transformations are also useful in an application concerning reference point programming as discussed in Chapter 9 of this volume. In particular, consider the problem

$$\begin{aligned} \text{P: Min } z &= \sum_j c_j \mid g_j - x_j \mid \\ \text{s.t. } & \mathbf{Ax} \le \mathbf{b} \\ & \mathbf{x} \ge \mathbf{0}. \end{aligned}$$

Defining new variables $y_j := |g_j - x_j| \; \forall j$, we can rewrite the problem as

$$\begin{aligned} \text{P: Min } z &= \sum_j c_j y_j \\ \text{s.t. } & y_j \ge g_j - x_j \quad \forall j \\ & y_j \ge x_j - g_j \quad \forall j \\ & \mathbf{Ax} \le \mathbf{b} \\ & \mathbf{x}, \mathbf{y} \ge \mathbf{0}. \end{aligned}$$

As in the above transformation, the additional constraints that link the variables x_j and y_j ensure that y_j is at least as large as $|g_j - x_j|$, and the minimization function, coupled with nonnegative weights c_j, ensures that y_j will actually equal the absolute value. This is the desired result.

8.3.2 Bottleneck Problems

The first author to describe bottleneck problems appears to have been Barsow (1959). In this context, the term "bottleneck" is not to be understood as constraints that are tight at optimum as is customary, but in a much narrower context: Bottleneck programming problems are mathematical formulations with a special type of objective function that minimizes the maximal cost coefficient of any variable with strictly positive value. The set of constraints is the same as in any other optimization problem. In this section we will restrict ourselves to linear constraints. Formally, bottleneck programming problems with linear constraints. i.e. *bottleneck linear programming problems* (or *BLP* for short) can be written as:

$$\begin{aligned} \text{P}_{\text{maximin}}\text{: } \max_{\mathbf{x}} \; z &= \min_j \{c_j \text{: } x_j > 0\} \\ \text{s.t. } & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \ge \mathbf{0}, \end{aligned}$$

or

$$\begin{aligned} \text{P}_{\text{minimax}}\text{: } \min_{\mathbf{x}} \; z &= \max_j \{c_j \text{: } x_j > 0\} \\ \text{s.t. } & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \ge \mathbf{0}. \end{aligned}$$

A special case of bottleneck problems was first described by Fulkerson, Glicksberg and Gross (1953) and then by Gross (1959). Later a variety of new theoretical and methodological developments were made by Hammer (1969),

Edmonds and Fulkerson (1970), Garfinkel and Rao (1971) and (1976), Kaplan (1976) and Posner and Wu (1981), to name just a few.

Applications of bottleneck linear programming problems are found in areas such as political districting and location models. Two simple numerical problems will be formulated in the following

Example 1: A bank manager has to allocate \$200,000 to five investment alternatives. The investment in each of the alternatives is limited to no more than \$60,000. The probabilities for a total loss of the investments are 1%, 3%, 2%, 6% and 4%, respectively. Being very cautious, the manager wishes to limit minimize the maximal loss probability of any of the investment alternatives to which money is allocated.

Defining $x_j, j = 1,\ldots, 5$ to be the amount of money allocated to the j-th investment alternatives, the mathematical formulation is:

$$\text{P: } \min_{\mathbf{x}} \; z = \max_{j} \{c_j : x_j > 0\}$$

with $c_1 = .01$, $c_2 = .03$, $c_3 = .02$, $c_4 = .06$, and $c_5 = .04$,

$$\text{s.t. } x_1 + x_2 + x_3 + x_4 + x_5 = 200{,}000$$
$$x_j \le 60{,}000 \;\forall\, j = 1, \ldots, 5$$
$$x_1,\ x_2,\ x_3,\ x_4,\ x_5 \ge 0.$$

Alternatively, this minimax bottleneck linear programming problem could have been formulated with an objective that minimizes the expected or the worst-case loss; for formulations with the latter objective, see Section 8.3.3.

Example 2: A Federal Government agency has a budget of \$3,200,000 which may be spent on measures against the pollution of lakes. Assume in this hypothetical example that the budget cannot be overspent. There are five lakes with pollution levels of 2, 5, 7, 4 and 1, respectively and the planning agency has to decide which lakes to clean up, assuming that partial cleanups are not feasible or desirable. The cleanup costs for the lakes are \$700,000; \$1,200,000; \$1,600,000, \$1,100,000 and \$900,000 respectively. The objective of the government agency is to take antipollution measures, so that after the cleaning, the worst polluted lake is as clean as possible.

In order to model the above situation, we define zero-one variables $x_j, j = 1,\ldots, 5$, such that

$$x_j = \begin{cases} 1, \text{ if the } j\text{ - th lake is not cleaned} \\ 0 \text{ otherwise} \end{cases}.$$

The budget constraint (in millions of dollars) can be written as $0.7(1 - x_1) + 1.2(1 - x_2) + 1.6(1 - x_3) + 1.1(1 - x_4) + 0.9(1 - x_5) \leq 3.2$ or $.7x_1 + 1.2x_2 + 1.6x_3 + 1.1x_4 + 0.9x_5 \geq 2.3$, so that the model can be written as

$$\text{P: Min } z = \max_j \{c_j : x_j > 0\}$$
$$\text{with } c_1 = .02,\ c_2 = .05,\ c_3 = .07,\ c_4 = .04,\ c_5 = .01$$

$$\text{s.t.} \quad 7x_1 + 12x_2 + 16x_3 + 11x_4 + 9x_5 \geq 23$$
$$x_1,\ x_2,\ x_3,\ x_4,\ x_5 = 0 \vee 1.$$

Example 3: Bottleneck assignment problems are defined as models with the usual assignment constraints that have a bottleneck objective function. The reasoning behind a bottleneck objective in the context of an assignment problem can be envisaged as follows. Consider the usual worker-machine assignment scenario in which a part is processed by workers assigned to machines along an assembly line. The coefficient c_{ij} denotes the quality of the work performed on the part by the i-th worker assigned to the j-th machine. The underlying assumption of the bottleneck objective in this example is that the quality of the final product, i.e., the part after it has been processed by all workers and machines, is just as good as the lowest quality job performed on it. In more popular terms, a chain is not stronger than its weakest link. To give a numerical example where high numbers denote a high quality, suppose that a particular worker-machine assignment in a 5 worker, 5 machine setting results in qualities 6, 6, 6, 6 and 6 for the various jobs, and a different assignment of the same set of workers to the same set of machines results in qualities 10, 10, 9, 4 and 10. Then the first worker-machine assignment would be preferred since the quality of all jobs is 6 whereas in the second case it would only be 4.

Similar to our discussion in linear programming, it is again useful to determine the set of solutions that have the same value of the objective function. In standard linear programming, such a set determined an iso-profit line. In contrast, the set of points with equal objective value in the case of bottleneck problems are highly discontinuous. As an example, consider the minimax bottleneck problem Min $z = \max_{x_1, x_2 > 0} \{c_1, c_2\}$ with $c_1 = 3$ and $c_2 = 5$. The entire space can then be subdivided into three regions, each of which has solutions of equal objective value. The origin has a value of $z = 0$, all points with $x_1 > 0$ and $x_2 = 0$ have $z = 3$, and all remaining points in the nonnegative orthant have an objective value of $z = 5$.

A useful property of bottleneck problems was proved by Garfinkel and Rao (1976) regarding optimal solutions. It is restated here as

Lemma 8.3: If a bottleneck linear programming problem has feasible solutions, then at least one of its optimal solutions is a basic solution.

The idea behind the proof is as follows. Assume that the variables have been renumbered, such that $c_1 \le c_2 \le c_3 \le \ldots \le c_n$ for minimax and $c_1 \ge c_2 \ge c_3 \ge \ldots \ge c_n$ for maximin bottleneck problems. Suppose that in a minimax bottleneck problem there are p basic solutions $\mathbf{x}^1$, $\mathbf{x}^2$, …, $\mathbf{x}^p$, with $x^1_{j_1}$, $x^2_{j_2}, \ldots, x^p_{j_p}$ being the variable with the highest subscript that has a positive value in each of the p solutions. Without loss of generality let $j_1 \ge j_2 \ge \ldots \ge j_p$, so that $\bar{z} = c_p$. Any solution that is strictly better than the p basic solutions above will have to have $x_\ell = 0$ for $\ell = j_p$, …, n. However, since any solution that is not a basic solution can be expressed as a linear convex combination of basic solutions and a basic solution will have $x_\ell = \sum_k \lambda_k x^k_{j_k}$ which, for $0 < \lambda_k < 1$ and $\sum_k \lambda_k = 1$ has $x_\ell > 0$, contradicting that the new solution is strictly better than any of the existing solutions $\mathbf{x}^k$, $k = 1, \ldots, p$.

The reasoning behind the lemma allows us to use a sequential linear programming approach. This could be accomplished as follows (for the minimax bottleneck problem). Start with a problem that includes all constraints and only variable x_1. If there is a feasible solution, it is optimal with the objective value $\bar{z} = c_1$. If not, include variable x_2 as well. Again, if there is a feasible solution, it is optimal and the objective value is $\bar{z} = c_2$. Continue in this fashion until a solution is found or it has been determined that no feasible solution exists.

This procedure can be incorporated in a version of the standard simplex method that is identical to the first phase of the two-phase method, except for the pivot column selection rule. Adding slack, excess, and artificial variables A_i as usual, the artificial objective function is then $\text{Min } w_0 = \sum_i A_i = \sum_{j=1}^{n} w_j x_j$. The bottleneck simplex method can then be described as follows.

The Bottleneck Simplex Method (minimax)

Step 1: Is $w_0 = 0$?
If yes: Stop, the current solution is optimal
If no: Go to step 2.

Step 2: Is $w_j \ge 0 \ \forall \ j$?
If yes: Stop, the problem has no feasible solution
If no: Go to step 3.

Step 3: Determine $s = \min \{j: w_j < 0\}$; then the s-th column is the pivot column. The r-th row is chosen as pivot row, such that $\frac{b_r}{a_{rs}} = \min_{i:a_{is}>0}\left\{\frac{b_i}{a_{is}}\right\}$ and a_{rs} is the pivot element. Perform one tableau transformation with the pivot a_{rs} and go to Step 1.

The maximin version of the problem can be dealt with in a similar fashion. Note that the solution procedure does not require knowledge of the numerical values of the parameter c_j, only their ranking is relevant.

Example: Consider the following minimax bottleneck problem

$$\text{P: Min } z = \max_j \{c_j : x_j > 0\}$$

$$\begin{aligned} \text{s.t.} \quad & -3x_2 + 2x_3 + 3x_4 + 4x_5 + 2x_6 \le 24 \\ & -x_1 + 4x_3 - 2x_4 + 2x_5 + x_6 = 8 \\ & 2x_1 - x_2 + 3x_4 - x_6 \ge 3 \\ & -3x_1 + x_2 + x_3 - 2x_4 + x_5 + 4x_6 = 12 \\ & x_1, \; x_2, \; x_3, \; x_4, \; x_5, \; x_6 \ge 0. \end{aligned}$$

Adding variables S_1, A_2, $-E_3$, A_3, and A_4 to the left-hand sides of the respective constraints, the initial tableau is

T^1:

	x_1	x_2	x_3	x_4	x_5	x_6	S_1	A_2	E_3	A_3	A_4	1
	0	–3	2	3	4	2	1	0	0	0	0	24
	–1	0	(4)	–2	2	1	0	1	0	0	0	8
	2	–1	0	3	0	–1	0	0	–1	1	0	3
	–3	1	1	–2	1	4	0	0	0	0	1	12
aof	2	0	–5	1	–3	–4	0	0	1	0	0	–23

The eligible pivot columns are those belonging to the variables x_3, x_5, and x_6, and we choose the variable with the smallest subscript as the variable to enter the basis, here x_3. The pivot row, and with it the leaving variable, is determined by means of the usual "smallest ratio" rule in the simplex method, which here turns out to be the second row on the tableau. This means that the artificial variable A_2 will leave the basis, and we will delete it from the next tableau T^2.

T^2:

	x_1	x_2	x_3	x_4	x_5	x_6	S_1	E_3	A_3	A_4	1
	$\frac{1}{2}$	–3	0	4	3	$\frac{3}{2}$	1	0	0	0	20
	$-\frac{1}{4}$	0	1	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{4}$	0	0	0	0	2
	2	–1	0	(3)	0	–1	0	–1	1	0	3
	$-\frac{11}{4}$	1	0	$-\frac{3}{2}$	$\frac{1}{2}$	$\frac{15}{4}$	0	0	0	1	10
aof	$\frac{3}{4}$	0	0	$-\frac{3}{2}$	$-\frac{1}{2}$	$-\frac{11}{4}$	0	1	0	0	–13

Since $w_0 = -13 \neq 0$, the current solution is not yet feasible. The pivot-eligible column with the smallest subscript is the column that belongs to the variable x_4, which is chosen as the entering variable. The third row is the pivot row, so that the

artificial variable A_3 will leave the basis and it is deleted from further consideration. The next tableau is then T^3.

T^3:

	x_1	x_2	x_3	x_4	x_5	x_6	S_1	E_3	A_4	1
	$-\frac{13}{6}$	$-\frac{5}{3}$	0	0	3	$\frac{17}{6}$	1	$\frac{4}{3}$	0	16
	$\frac{1}{12}$	$-\frac{1}{6}$	1	0	$\frac{1}{2}$	$\frac{1}{12}$	0	$-\frac{1}{6}$	0	$\frac{5}{2}$
	$\frac{2}{3}$	$-\frac{1}{3}$	0	1	0	$-\frac{1}{3}$	0	$-\frac{1}{3}$	0	1
	$-\frac{7}{4}$	$\frac{1}{2}$	0	0	$\frac{1}{2}$	$\frac{13}{4}$	0	$-\frac{1}{2}$	1	$\frac{23}{2}$
aof	$\frac{7}{4}$	$-\frac{1}{2}$	0	0	$-\frac{1}{2}$	$-\frac{13}{4}$	0	$\frac{1}{2}$	0	$-\frac{23}{2}$

The solution in tableau T^3 is still not feasible. The pivot-eligible variable with the smallest subscript is x_2, so this variable will enter the basis. The only possible pivot is in the fourth row, so that the artificial variable A_4 will leave the basis, so that we can delete its column. (Note that it is not always the case that an artificial variable leaves the basis in each step). The next tableau is T^4.

T^4:

	x_1	x_2	x_3	x_4	x_5	x_6	S_1	E_3	1
	–8	0	0	0	$\frac{14}{3}$	$\frac{41}{3}$	1	$-\frac{1}{3}$	$54\frac{1}{3}$
	$-\frac{1}{2}$	0	1	0	$\frac{2}{3}$	$\frac{7}{6}$	0	$-\frac{1}{3}$	$6\frac{1}{3}$
	$-\frac{1}{2}$	0	0	1	$\frac{1}{3}$	$\frac{11}{6}$	0	$-\frac{2}{3}$	$8\frac{2}{3}$
	$-\frac{7}{2}$	1	0	0	1	$\frac{13}{2}$	0	–1	23
aof	0	0	0	0	0	0	0	0	0

It is apparent that the solution in tableau T^4 is feasible, so that the algorithm terminates with an optimal bottleneck solution. The optimal solution is $\bar{\mathbf{x}} = [0, 23, 6\frac{1}{3}, 8\frac{2}{3}, 0, 0]^T$ with a value of the objective function of $\bar{z} = c_4$.

A second example is a bottleneck assignment problem that demonstrates the threshold technique, a special case of the simplex bottleneck method.

Example: Consider a scenario in which five employees are to be assigned to five positions, so that each employee occupies exactly one position and each position is occupied by exactly one employee, i.e., the structure of the standard assignment problem. In the following matrix $\mathbf{A} = (a_{ij})$, an element a_{ij} denotes the quality of the job employee i does when assigned to position j. The main assumption is that the overall quality of the work is the lowest quality achieved by any of the employees in their position. This implies the obvious objective to maximize the minimal quality. The estimated quality of individual job assignments is collected in the matrix

$$\mathbf{A} = \begin{bmatrix} 3 & 2 & 1 & 2 & 4 \\ 5 & 3 & 2 & 4 & 4 \\ 4 & 5 & 3 & 1 & 2 \\ 1 & 4 & 6 & 4 & 2 \\ 2 & 4 & 5 & 6 & 6 \end{bmatrix},$$

where a high number indicates a high quality. The method will now try to determine a feasible assignment that uses only the highest values, which in this case is "6." However, it is apparent that none of the employees will attain this quality if assigned to position 1. Hence we lower our expectation and consider all assignments in which an employee-position assignment results in a quality of at least "5," the next lower assignment figure. This is accomplished by solving a regular (minsum "cost") assignment problem using a matrix $\mathbf{M} = (m_{ij})$, in which $m_{ij} = 0$ indicates an assignment that results in a quality of at least 5, while a "×" indicates that the assignment will fall short of the desired quality of "5." Here,

$$\mathbf{M} = \begin{bmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ \times & 0 & \times & \times & \times \\ \times & \times & 0 & \times & \times \\ \times & \times & 0 & 0 & 0 \end{bmatrix}.$$

As there is no feasible assignment in the first row (i.e., the first employee is not capable of achieving a quality of at least 5 in any of the positions), we lower the quality to the next level, which is 4. Redefining the matrix **M**, in which zeroes now indicate assignments with a quality ranking of at least 4, we find

$$\mathbf{M} = \begin{bmatrix} \times & \times & \times & \times & 0 \\ 0 & \times & \times & 0 & 0 \\ 0 & 0 & \times & \times & \times \\ \times & 0 & 0 & 0 & \times \\ \times & 0 & 0 & 0 & 0 \end{bmatrix}.$$

At this level, there is a variety of feasible (and hence optimal) solutions. The only really limiting assignment is that employee 1 must be assigned to position 5. Other than that, there is a variety of possible assignments. One such assignment has employees 1, 2, 3, 4, and 5 work in positions 5, 1, 2, 3, and 4, respectively. The job qualities in the individual assignments are 4, 5, 5, 6, and 6. It is obvious that employee 1 is the bottleneck. As a policy implication, the decision maker will consider additional training for that employee, as everybody else is able to perform higher-quality work.

The above procedure reveals that the optimal solution of an $[n \times n]$-dimensional bottleneck assignment problem can be determined by solving a sequence of regular minsum cost assignment problems. The number of such problems cannot be larger than the number of distinct values a_{ij}, which, in turn, is limited to n^2. Rather than starting from the highest possible quality and than stepping down as far as necessary one step at a time, bisection search could be used instead. Also note that a premature termination of the algorithm is not possible, the first feasible and potentially implementable solution is also optimal.

8.3.3 Minimax and Maximin Problems

The models discussed in the section have a certain resemblance to the bottleneck problems described previously. They also have a minimax or maximin objective function, but in the basic model presented here, the objective is to either minimize the maximal value of any of the given decision variables (maximize the minimum value of a decision variable), while bottleneck problems minimize (maximize) the attribute of the variable with the largest (smallest) subscript. Some relations between bottleneck problems and minimax formulations will be explored below.

As a motivation, consider a decision maker whose goal is to store ten tons of ammunition in four locations. The concern is that a direct hit may destroy some of the ammunition, so that the objective is to divide the ammunition, such that in case of a direct hit, the damage is minimized. Assuming that the decision maker wants to guard against the worst case, i.e., the highest potential loss, the problem is easily formulated by defining variables x_j that denote the amount of ammunition stored in location j, $j = 1, \ldots, 4$. We can then formulate the problem as

$$\begin{aligned} \text{P: Min } & z = \max \{x_j\} \\ \text{s.t. } & x_1 + x_2 + x_3 + x_4 = 10 \\ & x_1,\ x_2,\ x_3,\ x_4 \geq 0. \end{aligned}$$

Somewhat arbitrarily, consider now the three solutions $\mathbf{x}^1 = (10, 0, 0, 0)$, $\mathbf{x}^2 = (5, 0, 1, 4)$, and $\mathbf{x}^3 = (2.5, 2.5, 2.5, 2.5)$. Given the decision maker's pessimistic attitude (*viz*., that the worst case will occur, meaning that the location that houses the largest amount of ammunition, will get hit), the objective values of the three solution above are 10, 5, and 2.5, respectively, making $\mathbf{x}^3$ the best of the three solutions offered here. It is easy to see that this solution is also optimal, and that the minimax objective function has an equalizing effect. This is in marked difference to the usual minisum (or maxisum) objectives that tend of have "extreme" solution in the sense that typically only a few variables assume positive values, while many other variables equal zero. It is also apparent that the optimal solution of the minimax problem is not an extreme point of the polytope described by the constraints in the formulation. A similar formulation was put forward by Simmons (1972), who allocates money to insurance policies so as to guard against the worst possible loss.

Another application considers a system of m simultaneous linear equations in n variables, $\mathbf{Ax} = \mathbf{b}$, $\mathbf{x} \in \mathbb{R}^n$ which has no solution. It first introduces deviational variables $d_i \in \mathbb{R}$, $i = 1,\ldots, m$ which measure the difference between right- and left-hand sides. The objective is then to find a solution that is as close to feasibility as possible. One possible approach minimizes the maximal deviation of a solution to the feasible set. This minimax problem can be formulated as

$$\begin{aligned} \text{P: Min } z &= \max_i \{|\, d_i \,|\} \\ \text{s.t.} \quad & \mathbf{Ax} + \mathbf{d} = \mathbf{b} \\ & \mathbf{x} \in \mathbb{R}^n,\ \mathbf{d} \in \mathbb{R}^m. \end{aligned}$$

Another approach minimizes the sum of deviations from feasible solutions. This can be accomplished by solving the linear programming problem

$$\begin{aligned} \text{P: Min } z &= \sum_{i=1}^{m} |\, d_i \,| \\ \text{s.t. } & \mathbf{Ax} + \mathbf{d} = \mathbf{b} \\ & \mathbf{x} \in \mathbb{R}^n,\ \mathbf{d} \in \mathbb{R}^m. \end{aligned}$$

These problems will be modeled and solved in Chapter 9.3.3.

Applications of minimax problems in general are found in fields such as the routing towards emergency facilities, where the worst congestion in the system is to be minimized, or the design of sales districts, so as to maximize the district with the smallest number of potential customers. Generally speaking, minimax and maximin problems are applicable in those applications, in which it is desired to obtain balanced solutions.

Consider now the graphical representation of minimax, maximin, maximax, and minimin problems. Figures 8.3a and 8.3b indicate the gradients of the objective functions as well as some of the iso-profit lines (level curves).

First consider the simple minimax objective Min z = max $\{x_1, x_2\}$ and the maximax objective Max z = max $\{x_1, x_2\}$. For any fixed value of z, say $\hat{z}$, then $\hat{z} = \max\{x_1, x_2\}$ is the set of points on the two line segments between $\mathbf{x} = (\hat{z}, 0)$ and $\mathbf{x} = (\hat{z}, \hat{z})$ as well as between $\mathbf{x} = (\hat{z}, \hat{z})$ and $\mathbf{x} = (0, \hat{z})$. In Figure 8.3a, all points with z-values of 3, 4 and 5 are located on the northeast sides of the squares whose northeast corners are at $\mathbf{x} = (3, 3)$, $\mathbf{x} = (4, 4)$, and $\mathbf{x} = (5, 5)$, respectively. Note that the vertices of the angles of the iso-profit lines are located on the 45° line. The minimax objective now attempts move these angles as much into a southwesterly direction as possible, whereas the maximax objective would move the angles as much into a northeasterly direction as possible.

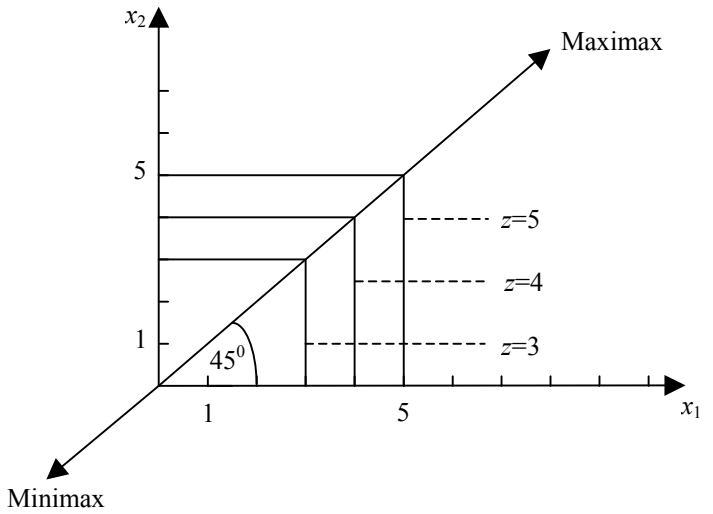


Figure 8.3a

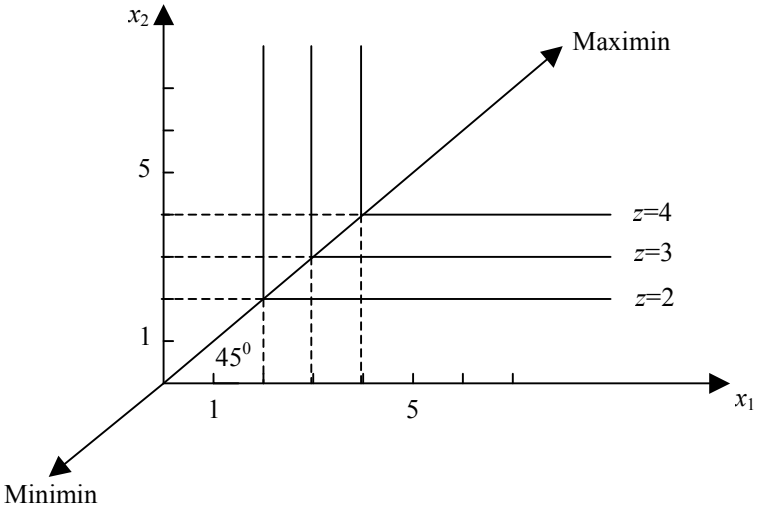


Figure 8.3b

On the other hand, consider the maximin objective Max z = min $\{x_1, x_2\}$ and the minimin objective Min z = min $\{x_1, x_2\}$. The iso-profit lines for these objectives are the angles with vertices at $\mathbf{x} = (\hat{z}, \hat{z})$ and their two sides parallel to the axes. Figure 8.3b displays set of points with $\hat{z}$ = 2, 3 and 4 respectively. The maximin objective now tries to move these angles as much as possible into a northeasterly

direction, while the minimin objective attempts to move them into a southwesterly direction.

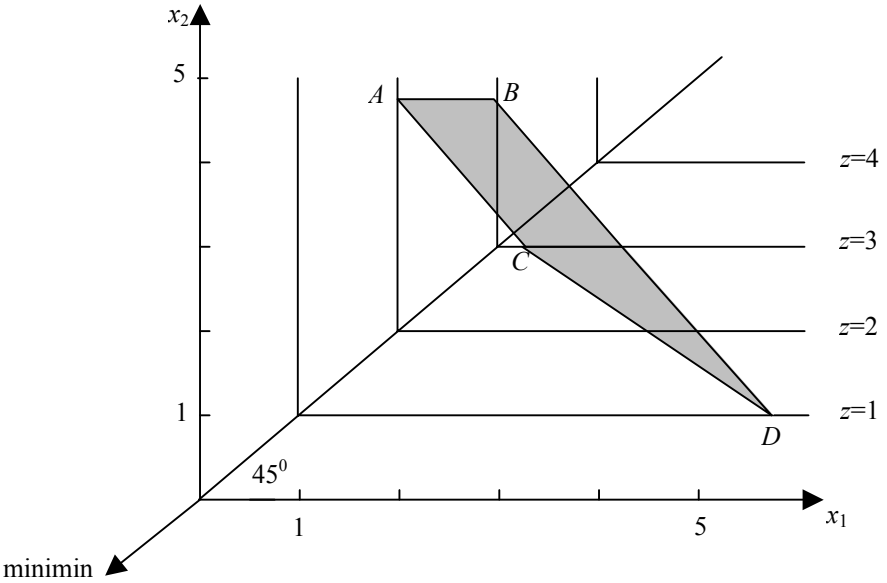


Figure 8.4a

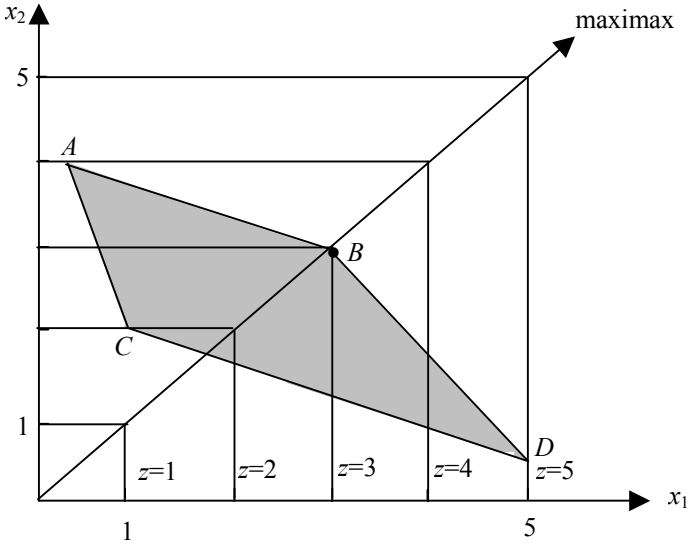


Figure 8.4b

One can easily show that $z = \max\{x_1, x_2\}$ is a convex function, whereas $z = \min\{x_1, x_2\}$ is concave. Therefore, the minimax and the maximin problems can be solved by means of feasible direction methods, whereas the minimin and maximax problems cannot. As an example, consider Figure 8.4a, where the shaded area indicates the feasible region. Assume that our starting solution is point A with its objective value $z = 2$ and suppose that we are using a minimin objective. Any move from the point A that stays within the feasible region and moves from A only some small distance $\varepsilon > 0$ will lead to solutions with higher, i.e., worse, values of the objective function. This could lead to the wrong conclusion that A is optimal, whereas the optimal solution is actually at point D where the objective value equals $\bar{z} = 1$. A similar argument can be made for the maximax objective in Figure 8.4b, where point A may mistakenly be considered optimal as all feasible points in its vicinity have objective values that are worse than that of A, which in reality, point D is optimal.

Due to the property of their objective functions, minimin and maximax problems cannot be transformed to and solved by a single linear programming problem each. It is, however possible to use a sequence of linear programming problems to solve these problems. More specifically, a minimin problem, formulated as P: Min $z = \min_{j=1,\ldots,n} \{x_j\}$, s.t. $\mathbf{Ax} \le \mathbf{b}$, $\mathbf{x} \ge \mathbf{0}$, could be replaced by a set of n linear programming problems P_j: Min $z_j = x_j$, s.t. $\mathbf{Ax} \le \mathbf{b}$, $\mathbf{x} \ge \mathbf{0}$, resulting in optimal solutions $\bar{\mathbf{x}}^j$, $j = 1,\ldots,n$. Now the k-th solution is optimal for the original problem P if $\bar{z}_k = \min_j \{\bar{z}_j\}$. Similarly, a maximax problem could be replaced by a sequence of linear programming problems.

On the other hand, minimax and maximin problems may be modeled as single linear programming problems. Rather than demonstrating the equivalence on a special problem, consider the more general formulation

$$\begin{aligned} \text{P: Min } z = & \max_{k=1,\ldots,r} \{\mathbf{c}^k\mathbf{x} + d_k\} \\ \text{s.t.} \quad & \mathbf{Ax} \le \mathbf{b} \\ & \mathbf{x} \ge \mathbf{0}, \end{aligned}$$

where the minimax function in the objective consists of r linear expressions $\mathbf{c}^k\mathbf{x} + d_k$ with $\mathbf{c}^k = [c_1^k, c_2^k, \ldots, c_n^k]$ and scalars d_k, $k = 1, \ldots, r$. Defining an $[r \times n]$-dimensional matrix $\mathbf{C} = \begin{bmatrix} \mathbf{c}^1 \\ \mathbf{c}^2 \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{c}^r \end{bmatrix}$ and an $[r+1]$-dimensional column vector $\mathbf{d} = (d_k)$, an

equivalent linear programming problem P′ can be defined using one additional variable z and r additional constraints as follows:

$$\begin{aligned} \text{P}'\text{: Min } & z \\ \text{s.t } & \mathbf{Ax} \qquad \leq \mathbf{b} \\ & \mathbf{Cx} - \mathbf{e}z \leq -\mathbf{d} \\ & \mathbf{x} \qquad \geq \mathbf{0}, z \in \mathbb{R}. \end{aligned}$$

Similarly, for a maximin problem with objective to Max $z = \min\limits_{k=1,\ldots,r} \{\mathbf{c}^k\mathbf{x} + d_k\}$ the equivalent linear programming formulation is P″: Max z, s.t. $\mathbf{Ax} \leq \mathbf{b}$, $\mathbf{Cx} - \mathbf{e}z \geq -\mathbf{d}$; $\mathbf{x} \geq \mathbf{0}$, $z \in \mathbb{R}$.

As an illustration, consider the following numerical

Example: Let a minimax problem be formulated as

$$\begin{aligned} \text{P: Min } z = \max & \{(-x_1 + 2x_2 + 14), (3x_1 - x_2 + 2)\} \\ \text{s.t.} \quad & x_1 + 2x_2 \leq 10 \\ & 4x_1 - 3x_2 \geq 6 \\ & x_1, \quad x_2 \geq 0. \end{aligned}$$

Here, $\mathbf{c}^1 = [-1, 2]$ and $\mathbf{c}^2 = [3, -1]$, so that

$$\mathbf{C} = \begin{bmatrix} c^1 \\ c^2 \end{bmatrix} = \begin{bmatrix} -1 & 2 \\ 3 & -1 \end{bmatrix} \text{ and } \mathbf{d} = \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} = \begin{bmatrix} 14 \\ 2 \end{bmatrix}.$$

Then the equivalent linear programming problem can be written as

$$\begin{aligned} \text{P}'\text{: Min } & z \\ \text{s.t.} \quad & x_1 + 2x_2 \qquad \leq 10 \\ & 4x_1 - 3x_2 \qquad \geq 6 \\ & -x_1 + 2x_2 - z \leq -14 \\ & 3x_1 - \ x_2 - z \leq -2 \\ & x_1, \quad x_2 \qquad \geq 0 \\ & \qquad\qquad z \in \mathbb{R}. \end{aligned}$$

The optimal solution is $\bar{x}_1 = 3$, $\bar{x}_2 = 0$, and $\bar{z} = 11$.

Figure 8.5 shows that the sets of points with fixed $z = \hat{z}$ value no longer form the corner of a right angle but some other angle with all vertices on a straight line which no longer leads through the origin.

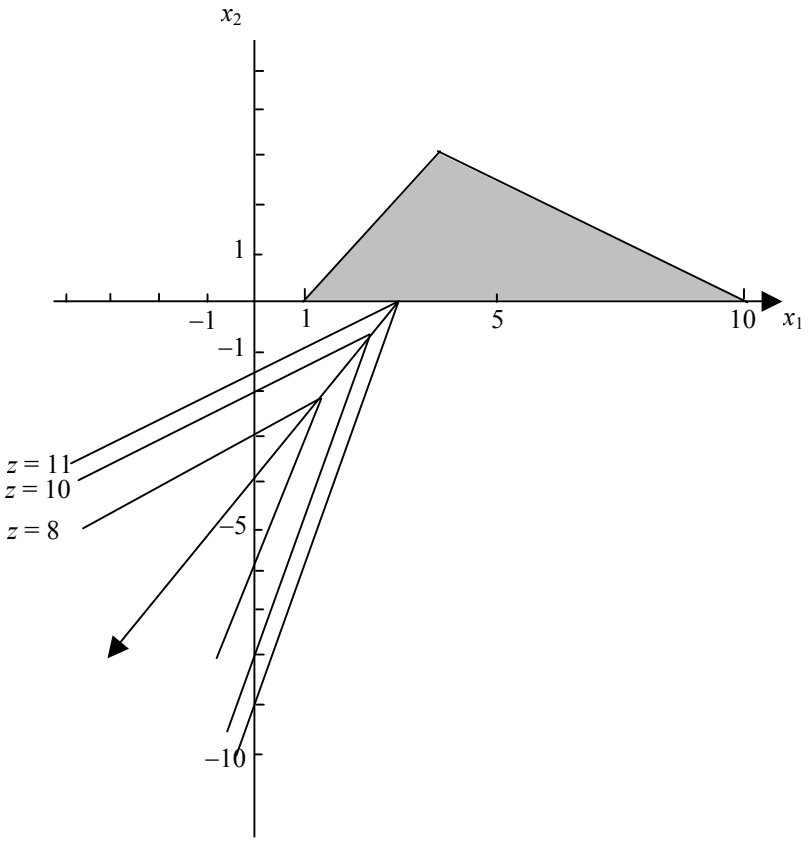


Figure 8.5

As already calculated algebraically, Figure 8.5 shows the optimal solution as well as the contour line associated with the optimal solution and some that have better values of the objective function, but are no longer feasible.

One final thought concerning relations between bottleneck problems and minimax problems should conclude this section. Consider the maximin bottleneck problem as defined in the previous section:

$$\begin{aligned} \text{P: } \operatorname*{Min}_{\mathbf{x}} \; & z = \max_{j} \{c_j : x_j > 0\} \\ \text{s.t. } & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}. \end{aligned}$$

Define then zero-one variables y_j and formulate the minimax problem

$$\begin{aligned} \text{P':}\ \min_{\mathbf{x}}\ & z = \max_{j} \{c_j y_j\} \\ \text{s.t.}\ & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \le M\mathbf{y} \\ & \mathbf{x} \ge \mathbf{0},\ \mathbf{y} \in \{0, 1\}^r. \end{aligned}$$

It is easy to demonstrate that problem P' solves the bottleneck problem P. Suppose that P has a solution $\bar{z} = c_k$, i.e., $\bar{x}_k > 0$ and $\bar{x}_j = 0\ \forall\ j > k$. Given that the minimax problem P' has the same constraints means that this solution will also be feasible for P'. It is clear that $\bar{x}_k > 0$ implies that $\bar{y}_k = 1$, while $\bar{y}_j$, could be either zero or one for all $j > k$. However, the minimization objective in P' forces y_j to zero for all $j > k$, establishing that the zero-one minimax problem P' can solve the bottleneck problem P.

8.3.4 Fractional (Hyperbolic) Programming

A useful transformation from a nonlinear to a linear programming problem is possible if the objective is fractional. This is commonly referred to as *fractional programming* or *hyperbolic programming*. Let the original problem be formulated as

$$\begin{aligned} \text{P: Max}\ z = & \frac{c_0 + \mathbf{cx}}{d_0 + \mathbf{dx}} \\ \text{s.t.}\ & \mathbf{Ax} \le \mathbf{b} \\ & \mathbf{x} \ge \mathbf{0}, \end{aligned}$$

where **c** and **d** are n-vectors, c_0 and d_0 are scalars, **A** is an $[m \times n]$-dimensional matrix, and **b** is an m-vector. Objectives of this type are manifold: it applies whenever one expression is not seen as absolute, but in relation to another measure. Typical examples are return on investment, profit per square foot (in the retail business), performance in relation to that of competitors, and others.

The nonlinear programming problem P can be linearized by first defining a single new variable $\lambda = [d_0 + \mathbf{dx}]^{-1} > 0$. The problem can then be rewritten as

$$\begin{aligned} \text{P':}\ \text{Max}\ & \lambda c_0 + \lambda \mathbf{cx} \\ \text{s.t.}\ & \lambda = [d_0 + \mathbf{dx}]^{-1} \\ & \mathbf{Ax} \le \mathbf{b} \\ & \mathbf{x} \ge \mathbf{0}. \end{aligned}$$

Multiplying $\mathbf{Ax} \le \mathbf{b}$ by λ and defining a new vector of variables $\mathbf{y} = \lambda\mathbf{x}$, the problem can be written as the linear programming problem

$$\text{P''}: \text{Max } \mathbf{cy} + c_o\lambda$$

$$\text{s.t. } \mathbf{Ay} - \mathbf{b}\lambda \le \mathbf{0}$$
$$\mathbf{dy} + d_o\lambda = 1$$
$$\mathbf{y} \ge \mathbf{0}$$
$$\lambda > 0.$$

Example: Consider the fractional programming problem

$$\text{P}: \text{ Max } z = \frac{2 + x_1 + 3x_2}{1 + 2x_1 + x_2}$$

$$\text{s.t. } x_1 + x_2 \le 2$$
$$x_1 \le 1$$
$$x_1,\ x_2 \ge 0.$$

With $\lambda = [1 + 2x_1 + x_2\ ]^{-1}$, $y_1 = \lambda x_1$ and $y_2 = \lambda x_2$, we obtain

$$\text{P''}: \text{Max } z = y_1 + 3\ y_2 + 2\lambda$$

$$\text{s.t. } y_1 + y_2 - 2\lambda \le 0$$
$$y_1 - \lambda \le 0$$
$$2y_1 + y_2 + \lambda = 1$$
$$y_1,\ y_2 \ge 0$$
$$\lambda > 0.$$

The optimal solution of problem P'' is $\bar{y}_1 = 0$, $\bar{y}_2 =$ ⅔ and $\bar{\lambda} =$ ⅓, and hence the solution to the original problem P can be reconstructed as $\bar{x}_1 = 0$, $\bar{x}_2 = 2$, and and $\bar{z} = 2$⅔.

For a more general and in-depth treatment of fractional programming and for a bibliography, see Frenk and Schaible (2004).

A popular application of this transformation is used to derive the linear programming problem that was introduced in the section on data envelopment analysis (for details, readers are referred to Section 2.5). To reiterate, there are m branches whose efficiency is to be determined with respect to n^+ outputs and n^- input factors. Here, the parameters $a^+_{i\ell}$ and $a^-_{k\ell}$ denote the output of good i produced by branch ℓ and the input of factor k required by branch ℓ, respectively. Define now weights u_i and v_k that are associated with the output i and the input k, respectively. The relative efficiency of some branch j can then be written as

$$E_j = \frac{\sum_i a_{ij}^+ u_i}{\sum_k a_{kj}^- v_k}.$$

Suppose now that it is our objective to evaluate the efficiency of branch j. It is then required to normalize the efficiencies of all other branches $\ell \neq j$, so as to guarantee bounded solutions. The problem can be written as

$$\begin{aligned}
&\text{Max } E_j = \frac{\sum_i a_{ij}^+ u_i}{\sum_k a_{kj}^- v_k} \\
&\text{s.t. } \frac{\sum_i a_{i\ell}^+ u_i}{\sum_k a_{k\ell}^- v_k} \leq 1 \quad \forall\ \ell \\
&\qquad u_i \geq 0\ \forall\ i,\ v_k \geq 0\ \forall\ k.
\end{aligned}$$

Multiplying the constraints by their respective denominators and moving all variables to the left-hand sides results in

$$\sum_i a_{i\ell}^+ u_i - \sum_k a_{k\ell}^- v_k \leq 0 \quad \forall\ \ell.$$

Defining a new variable $\lambda = \left[\sum_k a_{kj}^- v_k\right]^{-1} > 0$, we can then replace the denominator in the objective function by λ. Multiplying the constraints with λ, we obtain the equivalent problem

$$\begin{aligned}
&\text{Max } E_j = \sum_i a_{ij}^+ u_i \lambda \\
&\text{s.t.} \quad \sum_i a_{i\ell}^+ u_i \lambda - \sum_k a_{k\ell}^- v_k \lambda \leq 0 \quad \forall\ \ell \\
&\qquad \sum_k a_{kj}^- v_k \lambda = 1 \\
&\qquad u_i \geq 0\ \forall\ i,\ v_k \geq 0\ \forall\ k,\ \lambda > 0.
\end{aligned}$$

Defining new variables $u_i' = u_i\lambda$ and $v_k' = v_k\lambda$, this problem can be written as

$$\begin{aligned}
&\text{Max } E_j = \sum_i a_{ij}^{+} u_i' \\
&\text{s.t.} \quad \sum_i a_{i\ell}^{+} u_i' - \sum_k a_{k\ell}^{-} v_k' \le 0 \ \ \forall \ \ \ell \\
&\qquad \sum_k a_{kj}^{-} v_k' = 1 \\
&\qquad u_i' \ge 0 \ \forall \ i, \ v_k' \ge 0 \ \forall \ k.
\end{aligned}$$

Assigning now dual variables $w_\ell \ge 0$ to the first set of constraints and a single dual variable $E_j^D \in \mathbb{R}$ to the last constraint, the dual of this problem can then be written as

$$\begin{aligned}
&\text{Min } \ E_j^D \\
&\text{s.t.} \ \sum_\ell a_{i\ell}^{+} w_\ell \ge a_{ij}^{+} \quad \forall \ \ i \\
&\quad -\sum_\ell a_{k\ell}^{-} w_\ell + a_{kj}^{-} E_j^D \ge 0 \ \ \forall \ \ k \\
&\text{(or, equivalently, } \sum_\ell a_{k\ell}^{-} w_\ell \le a_{kj}^{-} E_j^D \ \ \forall \ \ k \text{)} \\
&\qquad w_\ell \ge 0 \ \forall \ \ell, \ E_j^D \in \mathbb{R}.
\end{aligned}$$

This is the problem we solve in order to determine the efficiency of the *k*-th branch, and it is identical to the problem formulated in Section 2.5.

It should be pointed out that it would be equally possible to minimize the input per unit of output instead of maximizing the inverse expression as was done here. An example of data envelopment analysis is provided in Section 2.5.

9 MULTIOBJECTIVE PROGRAMMING

In the mathematical optimization problems discussed so far, we have assumed that there was a single objective function. This may be valid in some circumstances (e.g., when a well-established company attempts to maximize its profit), it will be unsatisfactory in others. In particular, when there is more than a single stakeholder as is the case in virtually all public decision making problems, or more than a single concern exists. Typical concerns include some often interrelated measures such as profits, costs, revenues, market shares, and return on investment, but can also include other issues such as risk, pollution, the loss of property values, and other criteria. The former, sometimes referred to as "private objectives" tend to be easily measured, while the latter, also referred to as "public objectives," are typically difficult to measure. For instance, pollution is almost always an expression that combines different pollutants that are not difficult to measure individually, while modeling their aggregation is certainly problematic. For simplicity, we will assume throughout this chapter that r issues of concern have been identified and agreed upon by all decision makers, and that these concerns can be expressed by means of a quantitative measure.

Before progressing any further, let us introduce some terminology. In general, we speak of *multi-criteria-decision making* or *MCDM* whenever multiple concerns (objectives or criteria) exist. The field of multi-criteria decision making is then commonly subdivided into *multi-attribute decision making* or *MADM* on the one hand and *multi-objective (linear) programming* or *MO(L)P* on the other. The difference is that in multi-attribute decision making, the decision maker is to choose between a finite number of already existing solutions, while multi-objective programming problems include a number of objectives that are to be optimized, typically in continuous space. In this chapter, we will restrict ourselves to multi-objective linear programming problems. For a summary of *MADM* models and techniques, see, e.g., Cohon (1978) or Eiselt and Sandblom (2004).

A brief account of the history of multiobjective optimization methods is provided by Ballestero and Romero (1998). The origins of the field can be traced back to Koopmans (1951a) and Kuhn and Tucker (1951). The former contribution introduces the notion of domination in the field, while the second paper develops

optimality conditions. Bicriterion models, i.e., models with two objectives were first discussed by Geoffrion (1967). A major impact was the first conference on multicriteria decision making at the University of South Carolina in 1972. The Proceedings of this conference were published by Cochrane and Zeleny (1973). After that milestone, activity in the field increased tremendously, witnessed by the thousands of references collected Stadler (1984) who already collected about 1,700 references in the mid-1980s. Today an internet search will result in no less than 240,000 hits when using "multiobjective optimization" as a search argument.

A seemingly obvious question in multiobjective optimization problems is whether individual issues should be expressed as constraints or objectives. At first glance, the distinction between objectives and constraints appears to be clear. Closer inspection will, however, reveal that while the formal distinction between objectives and constraints remains crisp, it is quite blurry as far as the applications are concerned. This concept may be explained by using an example from a survey by Eiselt and Laporte (1987). The issue is the scheduling of exams at a large University. Scheduling any two exams that are to be written by at least one student at exactly the same time is obviously not feasible and must be formulated as a constraint. Scheduling exams, such that a student will have to write a three-hour exam from, say 8 a.m. to 11 a.m. and another from 12 noon to 3 p.m. on the same day is highly undesirable, but possible. In order to avoid such incidents as much as possible, such schedules will carry a high penalty, but will not be prohibited by means of a constraint, as this is clearly a softer requirement as compared to the former case.

Even softer is to avoid schedules in which a student is scheduled for an exam on the first day of the exam period and another on the last day of the exam period. Again, there will be a penalty associated with such a schedule (albeit a smaller one as compared to the previous case), but again, it will not be prohibited. As a matter of fact, prohibiting all unpleasant occurrences by way of (absolute) constraints will typically result in the nonexistence of feasible solutions. In contrast, using penalties will result in solutions that may not be "pleasant" (more than one exam on a single day or exams for one student spaced throughout the exam period), but are balancing the unpleasantness among all students in the model.

With the distinction between objectives and constraints now being suitably blurred, we may now distinguish between different approaches on the basis of the decision maker's input. Assume that it has been decided which issues are modeled as (hard) constraints and which are expressed in (softer) objectives. We first distinguish between approaches in which the decision maker feels unable to provide any input (beyond establishing constraints and objectives), and those where he does.

9.1 Vector Optimization

The vector optimization problems discussed in this section assume no information on the part of the decision maker beyond the actual specification of the constraints and the objectives. In particular, vector optimization models are suited for cases in which the decision maker feels unable to specify tradeoffs between objectives or target values for the individual objectives. While it sounds attractive in the beginning, this model will result in a large number of solutions that will subsequently be evaluated by the decision maker. Thus, the need for input from the decision maker has not been eliminated but postponed.

One important feature of optimization models with multiple objectives is that the concept of optimality, in the way it was defined for single-objective optimization problems, does no longer apply. It is important to realize that a solution can only be called "optimal" with respect to one criterion or one objective. Hence it will be mandatory to generalize the concept of optimality. Usually, decision makers employ the concept of pareto-optimality first put forward by the Italian economist Pareto (1906). It essentially states that a solution or decision is called *pareto-optimal* (or, alternatively, *efficient*, *nondominated*, or *noninferior*), if there is no other feasible solution that is equal or better with respect to all objectives included in the model. The first to determine pareto-optimal solutions for multiobjective programming problem appears to have been Yu (1973). Formally, we can write

Definition 9.1: Given maximization objectives Max z_1, Max z_2, …, Max z_r, decisions d_1, d_2, …, d_ℓ, and profits (or payoffs) a_{ij} for decision d_i with respect to objective z_j, a decision d_i dominates a decision d_k, if $a_{ij} \geq a_{kj}\ \forall\ j$ with the inequality being strict for at least one j.

All dominated decisions can be deleted from further consideration. The decision maker will then make a choice among the remaining set of nondominated solutions.

As an illustration of the concept, consider the following

Example: Suppose that a decision maker has three objectives that are to be maximized, *viz*., Max z_1, Max z_2, and Max z_3, and assume that four solutions have been determined. The solutions achieve objective functions as shown in Table 9.1.

Table 9.1

	z_1	z_2	z_3
Solution 1	7	5	1
Solution 2	3	0	3
Solution 3	4	–1	8
Solution 4	6	2	3

Here, Solution 1 is better than Solution 2 with respect to the first two objectives, as its achievements are 7 and 5 and thus are considerably higher than that of Solution 2 whose achievements are only 3 and 0, but on objective 3, Solution 1 achieves only 1 whereas Solution 2 achieves 3, so that there is no dominance. In this example, only Solution 4 dominates Solution 2 as it its achievements are the same or higher with respect to all three objectives. Consequently, a rational decision maker will never choose Solution 2, which can be deleted from consideration. Among the other three solutions, there is no clear dominance, so that all three of them will be called efficient.

Given that not all objectives can usually be optimized at the same time, we will follow a common convention and write "Max" z rather than Max z. We can then formulate the following vector optimization problem as

$$\begin{aligned} \text{P: ``Max''} \quad & z_1 = \mathbf{c}^1\mathbf{x} \\ & z_2 = \mathbf{c}^2\mathbf{x} \\ & \cdots \\ & z_r = \mathbf{c}^r\mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \le \mathbf{b} \\ & \mathbf{x} \ge \mathbf{0}, \end{aligned}$$

where $\mathbf{c}^k$ denotes the coefficients of the k-th objective of the decision maker. A more compact form would assemble all objective function coefficients in the matrix $\mathbf{C} = \begin{bmatrix} \mathbf{c}^1 \\ \mathbf{c}^2 \\ \vdots \\ \mathbf{c}^r \end{bmatrix}$ and objective values in the vector $\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_r \end{bmatrix}$ so that the problem can be written as

$$\begin{aligned} \text{``Max''} \ \mathbf{z} &= \mathbf{C}\mathbf{x} \\ \text{s.t.} \quad \mathbf{A}\mathbf{x} &\le \mathbf{b} \\ \mathbf{x} &\ge \mathbf{0}. \end{aligned}$$

In order to investigate such a problem, consider first the objective functions and ignore the constraints. For our discussion, suppose that we have a point $\hat{\mathbf{x}}$ and two objective functions "Max" $z_1 = \mathbf{c}^1\mathbf{x}$ and "Max" $z_2 = \mathbf{c}^2\mathbf{x}$. The situation is displayed in Figure 9.1a, where the two objectives are anchored at the point $\hat{\mathbf{x}}$ and the lines (1) and (2) represent the iso-profit lines of the respective objective functions through the point $\hat{\mathbf{x}}$. The discussion of the graphical representation of linear programming has revealed that all points on line (1) are as good as $\hat{\mathbf{x}}$ with respect to the first objective, while all points on line (2) are as good as $\hat{\mathbf{x}}$ with respect to the second objective. Furthermore, all points in the halfspace indicated by two arrows pointing away from line (1) are better than $\hat{\mathbf{x}}$ with respect to the first

objective. Similarly, all points in the halfspace shown by the arrows on line (2) are better than point $\hat{\mathbf{x}}$ with respect to the second objective.

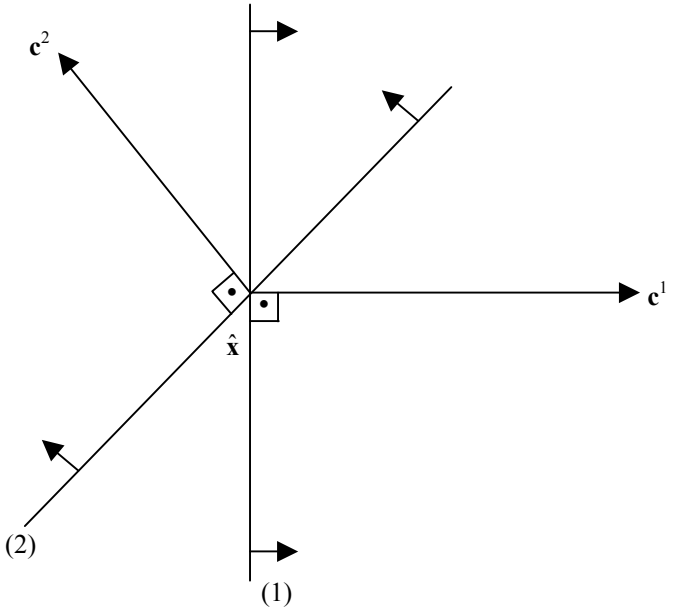


Figure 9.1a

For simplicity of the exposition Figure 9.1b shows the same situation again, but without the objectives, which are not necessary once the iso-profit lines through $\hat{\mathbf{x}}$ have been determined.

The two iso-profit lines through $\hat{\mathbf{x}}$ subdivide the plane into four cones, which have here been denoted by C^{--}, C^{+-}, C^{-+}, and C^{++}, respectively. Consider any point in the cone denoted by C^{--} and compare this point to the benchmark point $\hat{\mathbf{x}}$. Given its location, any point in C^{--} has a lower value of the first and the second objective as compared to the benchmark, thus the name of the cone. Clearly, whenever the benchmark point is realized, it would not make sense to move into the cone C^{--}, as both objective function values would deteriorate in such a move. Similarly, consider the cone C^{+-}. Given its location in relation to the iso-profit lines, any point in C^{+-} is better than the benchmark point $\hat{\mathbf{x}}$ with respect to the first objective, but worse than $\hat{\mathbf{x}}$ with respect to the second objective. Hence the comparison between any point in C^{+-} and the benchmark point $\hat{\mathbf{x}}$ is inconclusive. The same argument applies to points in the cone C^{-+}.

Consider now a point in the cone C^{++}. Given the location of the benchmark point $\hat{\mathbf{x}}$ in relation to any point in the cone C^{++} indicates that moving from $\hat{\mathbf{x}}$ into C^{++} will result in an improvement of both values of the objective function. This gives

the cone C^{++} its name *improvement cone*. The conclusion is here that from any given point $\hat{\mathbf{x}}$, we would like to move into the improvement cone defined by the objective functions, thus improving their values.

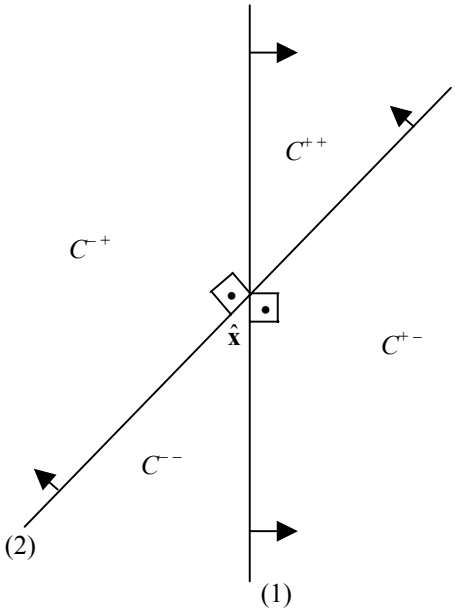


Figure 9.1b

Before integrating the above results with the consideration of feasible sets, we would like to point out that

- the angle between the gradients of any two objective functions is an indicator of the degree of conflict between the objectives, and
- the above arguments hold for any number of objectives.

First consider the angle between two objective functions. This angle is very small in Figure 9.2a, where the conflict is small (the two objectives are almost in agreement), while the situation in Figure 9.2b shows a large angle between the two objectives indicating extensive conflict. As the figure shows, the improvement cone in case of small conflict is large, while it is narrow when extensive conflict is present. Consider the two extreme cases: if the two objectives are identical, the improvement cone coincides with the halfspace generated by the objectives, as is the case in standard linear programming. On the other hand, if there is total conflict (i.e., the two objectives are diametrically opposed), the improvement cone degenerates to the empty set.

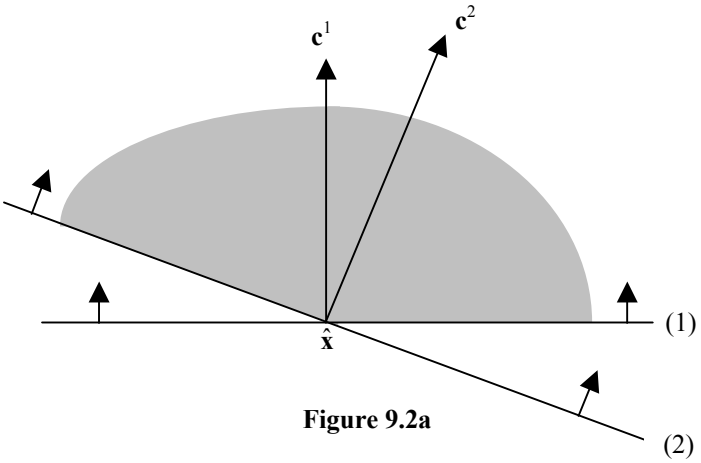


Figure 9.2a

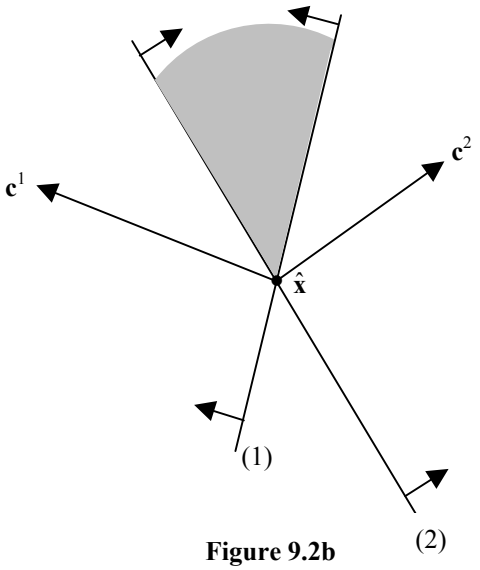


Figure 9.2b

Consider now the case of more than two objectives. The improvement cone is then the intersection of those halfspaces of the objectives that allow improvements (again shown by arrows at the iso-profit lines into the direction in which the objective value improves). In Figure 9.3, the first and third objectives generate the improvement cone that opens in the southwesterly direction, but when the second objective is introduced as well, the intersection of all three halfspaces is empty,

indicating that the improvement cone is empty, and that in this case, all points are nondominated.

In general, if the improvement cone has interior points, then the entire cone, including its boundary but excluding $\hat{\mathbf{x}}$ consists of points that are preferred to $\hat{\mathbf{x}}$. If, on the other hand, the improvement cone has no interior points, no points are better than $\hat{\mathbf{x}}$.

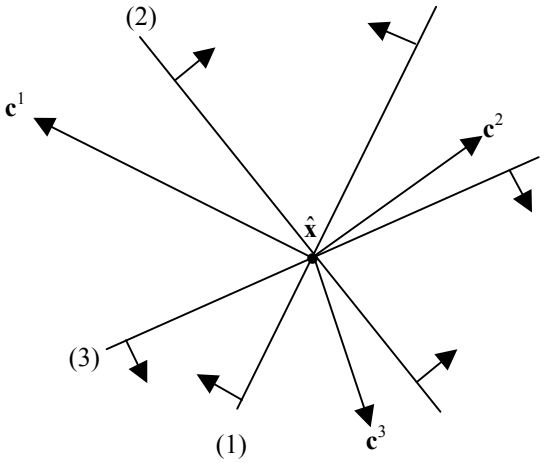


Figure 9.3

The visualization of multiple objectives and the concept of the improvement cone is also a valuable tool in the context of labor negotiations. Consider a highly simplistic situation, in which one decision maker represents labor, while the other represents management. Suppose that the only issue on the table is salary increases. Naturally, while labor wants to maximize salary increases, management will try to keep them as small as possible. In other words, there is total conflict. This situation is shown in Figure 9.4a.

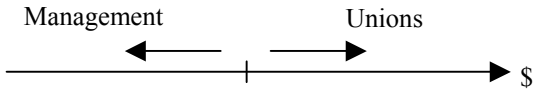


Figure 9.4a

Suppose now that the negotiators introduce a second issue, say, the introduction of additional paid coffee breaks, flexible working hours, additional options regarding retirement, or some other improvement of working conditions. The union and its

members will then decide on some tradeoff between the two issues. Clearly, salary increases will still be considerably more important than working conditions, but there will be a finite tradeoff. For management, the tradeoff between working conditions and additional labor costs is also finite. Actually, management might prefer the improvement of working conditions over a similar amount paid in additional salaries, as salaries may not improve productivity, while better working conditions could have that effect.

The objective of the two decision makers can then be visualized in Figure 9.4b. At this point we are also able to construct the improvement cone as introduced above, which is shown as C^{++}.

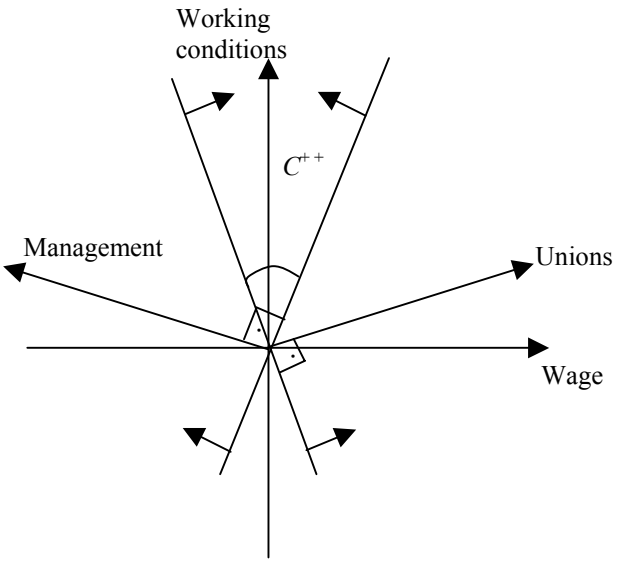


Figure 9.4b

It is now apparent that, in contrast to the original situation of total conflict, the improvement cone here will serve as a "compromise cone," in which the two decision makers could find solutions they can agree upon. The lesson here is that the introduction of additional noncontroversial issues may help find compromises.

We will now combine the results of the above discussion with the constraints and the feasible set. The basic idea is this. Suppose we would construct the improvement cone at any arbitrary feasible point $\hat{\mathbf{x}}$. Whenever it is possible to move from the point $\hat{\mathbf{x}}$ into the improvement cone and stay feasible, then the point under consideration cannot be nondominated. This clearly indicates that the set of nondominated points cannot include any interior points of the feasible set. On the other hand, any point on the boundary of the feasible set (i.e., not just

extreme points) may be noninferior. Because of the location of the noninferior points, the set of noninferior points is usually referred to as the *nondominated frontier*. Without proof, we state the following

Lemma 9.2: The nondominated frontier is a connected set.

To illustrate the concept, we put forward the following

Example: Consider the following vector optimization problem

$$
\begin{aligned}
\text{P: Max } & z_1 = -2x_1 + 3x_2 \\
\text{Max } & z_2 = 3x_1 - x_2 \\
\text{s.t.} \quad & -x_1 + x_2 \le 2 && (1)\\
& x_1 + 2x_2 \le 6 && (2)\\
& x_1 \le 4 && (3)\\
& x_1 + x_2 \ge 1 && (4)\\
& x_1, \; x_2 \ge 0.
\end{aligned}
$$

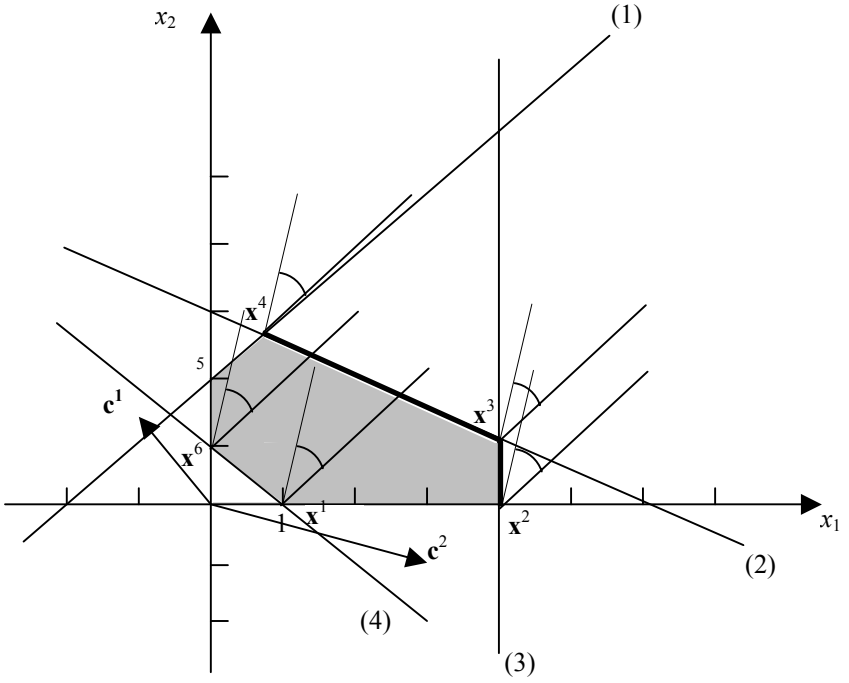


Figure 9.5a

The feasible set in Figure 9.5a is shaded and improvement cones have been constructed at all extreme points $\mathbf{x}^1 = (1, 0)$, $\mathbf{x}^2 = (4, 0)$, $\mathbf{x}^3 = (4, 1)$, $\mathbf{x}^4 = (2/3, 8/3)$, $\mathbf{x}^5 = (0, 2)$, and $\mathbf{x}^6 = (0, 1)$. It can be seen that the extreme points $\mathbf{x}^2$, $\mathbf{x}^3$, and $\mathbf{x}^4$ are all part of the nondominated frontier, as the intersection of the improvement cone at those points and the feasible set are only the points themselves, i.e., no improvement with respect to both objectives is possible. In case of the other three extreme points it is always possible to move into the improvement cone and stay in the feasible set, indicating their inferiority. The efficient frontier then includes the three nondominated extreme points as well as all boundary points between them. This is indicated in Figure 9.5a by the bold line segments between the extreme points $\mathbf{x}^2$ and $\mathbf{x}^3$ and between $\mathbf{x}^3$ and $\mathbf{x}^4$. The decision maker will then choose among the points on the efficient frontier.

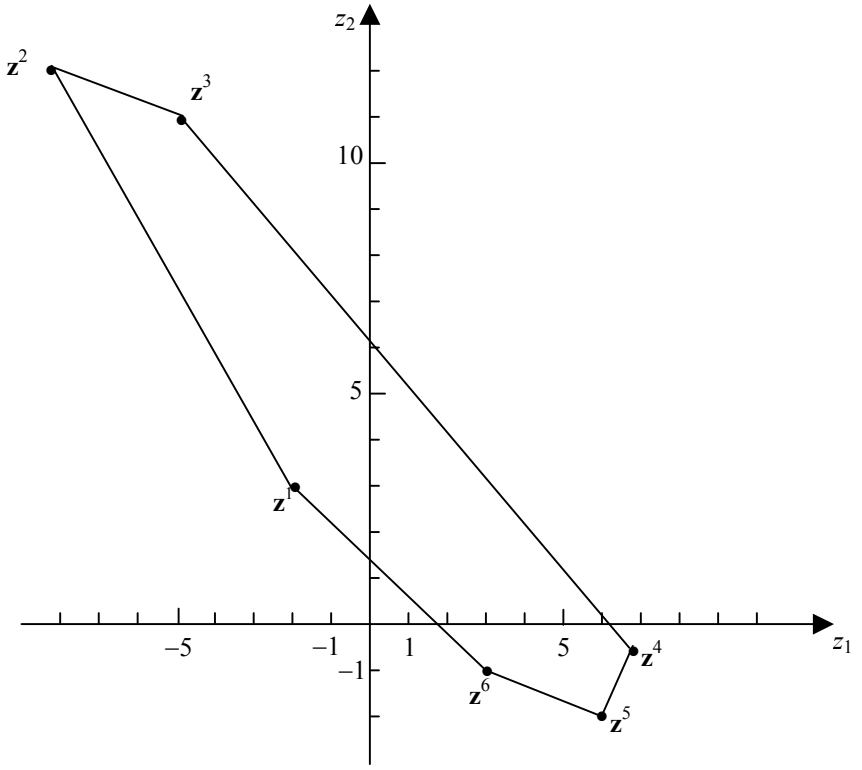


Figure 9.5b

Alternatively, we can plot the solutions in what is known as the *objective space*, in which the values of the objective functions are plotted along the axes and the extreme points are $\mathbf{z}^k = (z_1(\mathbf{x}^k), z_2(\mathbf{x}^k))$, i.e., the values of the objective function at the extreme points in the space of decision variables. In our example, $\mathbf{z}^1 = (–2, 3)$, $\mathbf{z}^2 = (–8, 12)$, $\mathbf{z}^3 = (–5, 11)$, $\mathbf{z}^4 = (20/3, –⅔)$, $\mathbf{z}^5 = (6, –2)$, and $\mathbf{z}^6 = (3, –1)$. The

feasible set in the objective space is shown in Figure 9.5b. The improvement cone in the objective space is simply the first quadrant shifted appropriately, which clearly indicates that among the six solutions, only $\mathbf{z}^2$, $\mathbf{z}^3$, and $\mathbf{z}^4$ are noninferior; the same result that we obtained earlier in the space of the decision variables.

At this point we have reduced the problem at hand to the task of finding the efficient frontier. Unfortunately, that task is **NP**-hard, as it may include an exponential number of extreme points, not to mention all linear convex combinations of adjacent efficient extreme points.

One possibility to generate all efficient extreme points is the *multiobjective simplex method*. The technique is due to Evans and Steuer (1973), Philip (1972) and Zeleny (1973). For a more recent account of the multiobjective simplex method, readers are referred to Ehrgott (2005). This is an exact method. Depending on the degree of conflict, it may take a very long time to determine all nondominated extreme points. As Ballestero and Romero (1998) report, the multiobjective simplex method has been used to solve problems to about 50 decision variables and three objective functions by the ADBASE program developed by Steuer (1995).

A variety of methods exist that approximate the efficient frontier. Most prominently among them are the *weighting method* and the *constraint method*. Their application does, however, require some additional input from the decision maker, so that they do not really belong in this category and we will therefore discuss them in Section 9.2.

In the special case of bi-objective problems, it is possible to construct *tradeoff curves* that are very helpful tools in the decision making process. In essence, they will plot the achievement of one objective function against the other in the (z_1, z_2) space. As such, they are the nondominated frontier in the objective space.

Ultimately, the trouble with this approach is that while the decision maker took the easy way out initially by not specifying any relation between the objectives, any external target values, and other information, there are now dozens (or, typically in the case of the multiobjective simplex method, hundreds of thousands) of solutions that the decision maker will have to compare, obviously an enormous task, even if only a few chosen solutions are to be compared. This is where additional input by the decision maker is required. This additional input can come either as a set of decision rules, tradeoffs, or similar information that is explicitly included in the model, or some interactive process. Some of these methods are described in the sections below.

9.2 Models with Exogenous Tradeoffs Between Objectives

This section discusses approaches to multiobjective optimization problems that use some input from the decision maker. Here, we consider only input that specifies relations between objectives and the choice of expressing a criterion as an objective or a constraint. Models that require input which specifies the relations between the achievement of a computed solution and some desired target value will be discussed in Section 9.3.

9.2.1 The Weighting Method

In case the decision maker is able to specify a finite tradeoff between any pair of objectives, it is possible to apply the *weighting method* that was suggested by Zadeh (1963). The basic idea of the weighting method is simple: Given the individual objectives Max $z_1 = \mathbf{c}^1\mathbf{x}$, Max $z_2 = \mathbf{c}^2\mathbf{x}$, …, Max $z_r = \mathbf{c}^r\mathbf{x}$, we use nonnegative weights w_1, w_2, …, w_r which are multiplied by the corresponding objective and then a composite objective is determined by adding up the weighted objectives. Formally, we construct the composite objective as Max $z = \sum_{k=1}^{r} w_k \mathbf{c}^k \mathbf{x}$. This objective is then repeatedly optimized for different combinations of weights.

If the decision maker is reasonably certain about the choice of weights, the optimization of the composite objective function with that weight combination will result in the desired combination (with appropriate sensitivity analyses using slightly perturbed weights following the optimization).

Most of the time the decision maker will not be able to specify precise weights. We can then use the weighting method to approximate the efficient frontier. Clearly, using a large number of objectives will require many combinations of weights to be examined. However, practical models will have rarely more than half a dozen or so objectives. Ideally, the analyst will start by optimizing one objective at a time, while ignoring all others, i.e., he will employ weight vectors $\mathbf{w}$ = [1, 0, …, 0], [0, 1, 0, …, 0], …, [0, 0, …, 0, 1] in r separate optimization runs. This will establish the extreme boundaries of the efficient frontier. Then, typically, in cooperation with the decision maker, the analyst will solve problems, in which the composite objective function is constructed by using weight combinations deemed reasonable by the decision maker. This alternating interactive process terminates when a solution is computed that the decision maker deems acceptable.

The weighting method can be illustrated by the following

Example: Consider the same bi-objective programming problem, which was introduced in Section 9.1. Optimizing the first objective (i.e., setting $\mathbf{w}$ = [1, 0]) results in the unique optimal solution $\bar{\mathbf{x}} = \left(\frac{2}{3}, \frac{8}{3}\right)$ with objectives values $(\bar{z}_1, \bar{z}_2) = \left(6\frac{2}{3}, -\frac{2}{3}\right)$, while optimizing the second objective results in the optimal

solution $\bar{\mathbf{x}} = (4,0)$ with objective values $(\bar{z}_1, \bar{z}_2) = (-8, 12)$. As can be seen in Figures 9.5a and 9.5b, these are indeed the two points at the end of the efficient frontier. The decision maker may now feel more comfortable, while not totally satisfied, with the former solution, so that the analyst may try a weight combination such as **w** = [.7, .3] or similar combinations. (It is common, but not necessary that the weights sum up to one). In Table 9.2, we provide optimal solutions for a number of weight combinations.

Table 9.2

Weights **w** = $[w_1, w_2]$	Composite objective	Optimal solution $\bar{\mathbf{x}} = (\bar{x}_1, \bar{x}_2)$	Optimal objective values $\bar{\mathbf{z}} = (\bar{z}_1, \bar{z}_2)$
1.0, 0	$-2x_1 + 3x_2$	$\left(\frac{2}{3}, \frac{8}{3}\right)$	$\left(6\frac{2}{3}, -\frac{2}{3}\right)$
.8, .2	$-x_1 + 2.2x_2$	$\left(\frac{2}{3}, \frac{8}{3}\right)$	$\left(6\frac{2}{3}, -\frac{2}{3}\right)$
.6, .4	$1.4x_2$	$\left(\frac{2}{3}, \frac{8}{3}\right)$	$\left(6\frac{2}{3}, -\frac{2}{3}\right)$
.5, .5	$.5x_1 + x_2$	$\left(\frac{2}{3}, \frac{8}{3}\right)$	$\left(6\frac{2}{3}, -\frac{2}{3}\right)$
.4, .6	$x_1 + .6x_2$	(4, 1)	(–5, 11)
.2, .8	$2x_1 - .2x_2$	(4, 0)	(–8, 12)
0, 1.0	$3x_1 - x_2$	(4, 0)	(–8, 12)

It turns out that as long as the weight of the first objective dominates the weight of the second objective, the point $\bar{\mathbf{x}} = (\bar{x}_1, \bar{x}_2) = \left(\frac{2}{3}, \frac{8}{3}\right)$ with objective values $\bar{\mathbf{z}} = (\bar{z}_1, \bar{z}_2) = \left(6\frac{2}{3}, -\frac{2}{3}\right)$ is optimal. In this example, it so happens that the weighting method generates all three efficient extreme points as we examine some possible weight combinations. In general, this will not be the case.

Furthermore, if the decision maker feels unable to choose between, say, the solutions $\bar{\mathbf{x}} = (\bar{x}_1, \bar{x}_2) = \left(\frac{2}{3}, \frac{8}{3}\right)$ and $\bar{\mathbf{x}} = (\bar{x}_1, \bar{x}_2) = (4, 1)$, but leans towards the latter, we can compute intermediate solutions by using weight combinations that favor the second point, such as $(\alpha_1, \alpha_2) = (.3, .7)$, so that the linear convex combinations of the two solutions is $(.3)\left(\frac{2}{3}, \frac{8}{3}\right) + (.7)(4, 1) = (3.0, 1.5)$ with the objective values $\bar{\mathbf{z}} = (\bar{z}_1, \bar{z}_2) = .3\left(6\frac{2}{3}, -\frac{2}{3}\right) + .7(-5, 11) = (-1.5, 7.5)$.

When using weights in the weighting method, consider the following. For simplicity, we assume again a bi-objective optimization model, i.e., a vector optimization model with just two objectives. The vector of weights **w** in the weighting method has two basic functions: first, the weights express the relative importance of the objectives, and secondly, they provide a means of conversion between the units in the different objectives, which may be incommensurable.

As an illustration, consider the following

Example: A decision maker wants to optimize some function of the two objectives profit and market share. The analyst has constructed the composite objective Max $z = w_1$(Profit) + w_2(Market share), which, by arbitrarily setting $w_1 = 1$ and using a weight $w = w_2$, results in the composite objective Max z = (Profit) + w(Market share). Suppose now that two distinct solutions have been determined, one with a profit of \$100,000 and a market share of 5%, and the other with \$50,000 profit and a 12% market share.

Here, the weight factor w denotes the conversion of 1% market share into an equivalent (profit) dollar value. Suppose now that a 1% market share is deemed to be worth \$10,000, then w = 10,000, and the two solutions have objective values of 100,000 + 10,000(5) = 150,000 and 50,000 + 10,000(12) = 170,000, so that the latter solution is preferred.

The *lexicographic method* assumes that the decision maker is able to specify a ranking of the objectives, such that the first objective is infinitely more important than the second objective, which, in turn, is infinitely more important than the third, and so forth. In that sense, the lexicographic method is nothing but a weighting method with infinite weights. Given such *preemptive priorities* (a very strong assumption, no doubt), it is possible to solve the problem by means of a sequence of linear programming problems. Assuming that the objectives are ranked from "Max" z_1 to "Max" z_2, …, "Max" z_r, we will ignore all objectives but the first. The resulting problem is a single-objective linear programming problem, which can be solved as usual. Suppose that the optimal value of the objective function is $\bar{z}_1$. We then add the constraint $\mathbf{c}^1\mathbf{x} \geq \bar{z}_1$ to the system of constraints and optimize the system with the objective Max $z_2 = \mathbf{c}^2\mathbf{x}$. The optimal solution of this system is then $\bar{z}_2$. We now add the constraint $\mathbf{c}^2\mathbf{x} \geq \bar{z}_2$ to the system and maximize the third objective. This procedure continues until all objectives have been optimized.

The drawback of this procedure is obvious: first, the decision maker has to make some very strong statements regarding the ranking of the objectives, and secondly, lower-ranking objectives are highly unlikely to be considered at all. For a good description, readers are referred to Collette and Siarry (2003). A variety of other techniques are discussed in Figueira *et al*. (2005).

9.2.2 The Constraint Method

The c*onstraint method* dates back to Marglin (1967). The basic idea is to transform all but one objective to constraints with unknown right-hand side values which represent different target values or achievement levels. For any given set of right-hand side values, the problem is but a standard linear programming problem. After the problem is solved for one set of achievement levels, their values are modified by the decision maker and the problem is solved again with a different

set of right-hand side values. This process is repeated until a solution has been found that is acceptable to the decision maker.

In order to illustrate the process, we use the same example that was employed to illustrate the weighting method. For convenience it is restated here.

Example: The bi-objective optimization model under consideration is

$$
\begin{aligned}
\text{P: Max } & z_1 = -2x_1 + 3x_2 \\
\text{Max } & z_2 = 3x_1 - x_2 \\
\text{s.t.} \quad & -x_1 + x_2 \le 2 && (1)\\
& x_1 + 2x_2 \le 6 && (2)\\
& x_1 \le 4 && (3)\\
& x_1 + x_2 \ge 1 && (4)\\
& x_1, \quad x_2 \ge 0.
\end{aligned}
$$

We arbitrarily choose the first objective to remain as an objective function, while the second objective is rewritten as a constraint with a preselected value of z_2, and the problem can then be written as

$$
\begin{aligned}
\text{P: Max } & z_1 = -2x_1 + 3x_2 \\
\text{s.t.} \quad & -x_1 + x_2 \le 2 && (1)\\
& x_1 + 2x_2 \le 6 && (2)\\
& x_1 \le 4 && (3)\\
& x_1 + x_2 \ge 1 && (4)\\
& 3x_1 - x_2 = z_2 && (5)\\
& x_1, \quad x_2 \ge 0.
\end{aligned}
$$

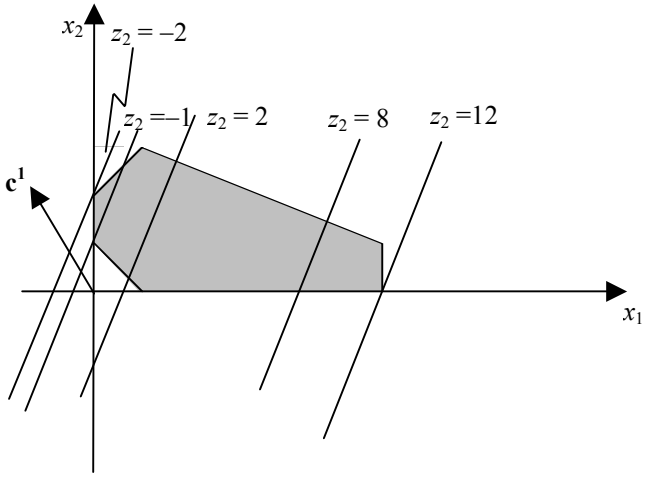


Figure 9.6

Figure 9.6 shows the feasible set defined by the constraints (1) – (4) and the nonnegativity constraints as the shaded area. In addition, there is the gradient of the objective function shown as $\mathbf{c}^1$ as well as contour lines for $z_2 = 12, 8, 2, -1,$ and -2. It can be seen that for z_2-values larger than 12 and smaller than -2, these contour lines do not intersect the feasible set, so that there exists no feasible solution.

The solutions obtained by the constraint method in the above example are summarized in Table 9.3 for a few selected values of z_2.

Table 9.3

z_2	Optimal solution $\bar{\mathbf{x}}$	Objective value $\bar{z}_1 =$
> 12	no feasible solution	—
12	(4, 0)	–8
8	(3.14, 1.43)	–2
4	(2, 2)	2
0	(.86, 2.57)	6
–1	(.5, 2.5)	5
–2	(0, 2)	6
<–2	no feasible solution	

As opposed to the weighting method, the constraint method typically does not find extreme points of the original feasible set.

Finally, we would like to mention *interactive methods* that may be used in order to incorporate the decision maker's input in a procedure that shuttles back and forth between the decision maker, who specifies some (typically limited) input and the analyst, who incorporates the decision maker's input in the model and resolves it. The earliest such methods date back to Benayoun *et al*. (1971) and the STEP method that uses a minimax objective to measure the weighted distance between the ideal point and the realized solution. Another early reference is Geoffrion *et al*. (1972). Reviews of interactive procedures are offered by Steuer (1986) and Shin and Ravindran (1991) and Hussein and Al-Ghaffar (1996). A good summary is provided by Reeves and Franz (1985) and recent references are Sakawa (2002) and Chen *et al*. (2005).

9.3 Models with Exogenous Achievement Levels

All models and approaches described in this section have in common that they necessitate information from the decision maker regarding the deviation of a solution from an exogenously specified measure. If this measure is some ideal (or reference) point, i.e., some ideal achievement level, then we deal with *reference*

point programming, as we will attempt to minimize the distance between the actual achievement level and that of the ideal point. Alternatively, reference points can also be least ideal (or nadir) points, so that the model will attempt to find a solution that is as far away from such a point as possible. In both cases, the decision maker must also specify a metric that is used to compute the distance between a solution and a reference point. A description of reference point methods in the discrete case can be found in Eiselt and Sandblom (2004), and a recent reference that applies the concepts is Buchanan and Gardiner (2003). Fuzzy programming is a concept in which desired achievements of the given objectives are assumed to be stated in an ambiguous way and the approach attempts to reconcile them. Finally, goal programming is a penalty method that works with target values and the method penalizes deviations from the prespecified targets.

9.3.1 Reference Point Programming

Reference point programming methods require that the decision maker specifies ideal or least ideal points for each of the objectives, as well as a metric that measures the distance between the actual achievement and the yardstick, defined by the (least) ideal point. Clearly, in case an ideal point is given, the distance between it and some solution generated by the solver is a disutility that is to be minimized, so that ideally, the ideal point is reached in which case the disutility (or penalty) equals zero. On the other hand, in case of a least ideal point, we would like to find solutions that are as far away as possible from the nadir point, so that the distance expresses a utility that we will attempt to maximize. Given the multiobjective programming problem

$$\begin{aligned} \text{P: ``Max''} \; & z_1 = \mathbf{c}^1\mathbf{x} \\ & z_2 = \mathbf{c}^2\mathbf{x} \\ & \dots \\ & z_r = \mathbf{c}^r\mathbf{x} \\ \text{s.t.} \quad & \mathbf{Ax} \le \mathbf{b} \\ & \mathbf{x} \ge \mathbf{0}, \end{aligned}$$

One way to determine an ideal point is to solve the r linear programming problems

$$\begin{aligned} \text{P: Max } z_k &= \mathbf{c}^k\mathbf{x} \\ \text{s.t.} \quad & \mathbf{Ax} \le \mathbf{b} \\ & \mathbf{x} \ge \mathbf{0}, \end{aligned}$$

and then define the ideal point as $(\bar{z}_1, \bar{z}_2, ..., \bar{z}_r)$. Choosing an ideal point that individually optimizes all objectives may not be a very good idea, as such a point may not properly reflect the decision maker's wishes.

To illustrate the concept, consider the example in Figure 9.7, in which the feasible set is plotted in the objective space (z_1, z_2). Minimizing the (Euclidean) distance from the ideal point to the set results in the point $\bar{\mathbf{z}}$ (note that $\bar{\mathbf{z}}$ is not an extreme point), while maximizing the (Euclidean) distance from the least ideal point to the set, we obtain the solution $\underline{\mathbf{z}}$. Intuitively, while $\bar{\mathbf{z}}$ appears to be a reasonable compromise solution, $\underline{\mathbf{z}}$ is not: while its achievement on the second objective is good, it fails miserably on the first objective with a comparatively high value of z_1. Worse yet, the point is dominated by one of its neighboring extreme points.

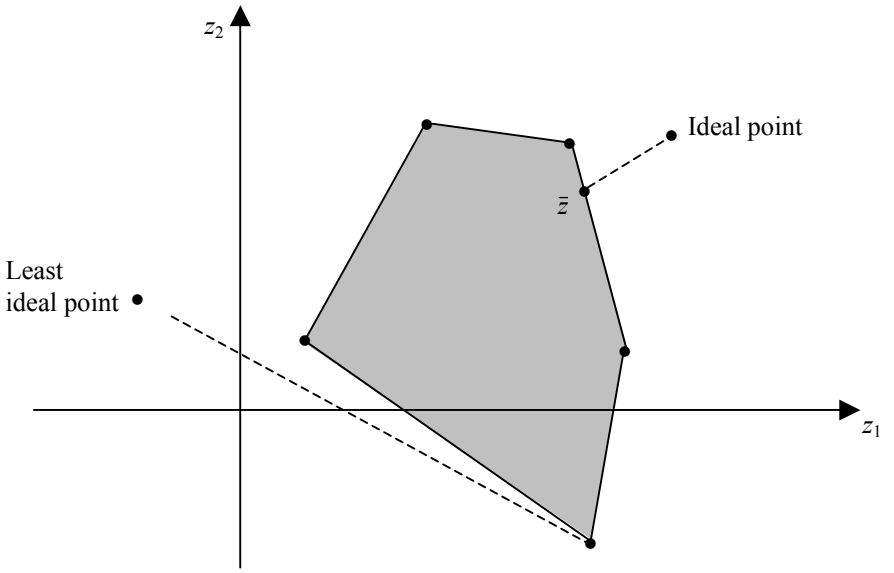


Figure 9.7

Another problem is the choice of distance function. Most authors will chose one among the class of *Minkowski metrics*. This is a class of distance functions that are define by a parameter p, so that the ℓ_p distance between two points $\mathbf{a} = (a_1, a_2, \ldots, a_n) \in \mathbb{R}^n$ and $\mathbf{b} = (b_1, b_2, \ldots, b_n) \in \mathbb{R}^n$ is defined as

$$d_p(\mathbf{a},\mathbf{b}) = \left[\sum_{k=1}^{n} |a_k - b_k|^p\right]^{\frac{1}{p}}.$$

For $p = 1$, we obtain the *Manhattan metric*

$$d_1(a,b) = \sum_{k=1}^{n} |\, a_k - b_k \,|,$$

i.e., the distance between two points is the sum of the coordinate differences. For $p = 2$, we obtain the usual *Euclidean* (or straight-line) *distances*

$$d_2(\mathbf{a}, \mathbf{b}) = \left[\sum_{k=1}^{n} (a_k - b_k)^2 \right]^{\frac{1}{2}},$$

which are obviously nonlinear. Finally, for $p \to \infty$, we obtain the *Chebyshev distances*

$$d_\infty(\mathbf{a}, \mathbf{b}) = \max_k \{|a_k - b_k|\}.$$

The Chebyshev distance is the largest individual absolute difference of any pair of coordinates between the two points and is useful in minimax settings. Note that Chebyshev distances are piecewise linear. As a matter of fact, only the ℓ_1 and ℓ_∞ distances lend themselves to formulations as linear programs.

Typically, the decision maker (or the analyst) will choose one of these metrics and apply it to the problem at hand. Zeleny's (1973) "compromise programming" is similar in that it first determines the nondominated set and the ideal point, which is the point that optimizes each objective individually. If the ideal point is feasible, this is the solution. Otherwise, the approach minimizes the deviation from the ideal point from the feasible set with some ℓ_p distance. The region that is optimal for any of these distances is the compromise region.

In a recent contribution, Buchanan and Gardiner (2003), minimize (maximize) Chebyshev distances between the feasible set and the ideal (least ideal) point.

We will illustrate the general idea by means of a numerical example that uses Manhattan distances.

Example: Consider a problem, whose feasible set is described by the constraints

$$\begin{aligned} x_2 &\leq 6 \\ x_1 + x_2 &\leq 8 \\ 2x_1 + x_2 &\leq 11 \\ x_1 &\leq 5 \\ x_1, \; x_2 &\geq 0. \end{aligned}$$

Suppose that the (unachievable) point (7, 4) has been identified by the decision maker as the ideal point. The task is now to define a distance function that allows us to measure the distance between the feasible set and the ideal point. The Manhattan distance $d_1(A, C)$ between two points $A = (a, b)$ and $C = (c, d)$ in the plane is then $d_1(A, C) = |a - b| + |c - d|$. The problem is now to minimize $|7 - x_1| + |4 - x_2|$ subject to the constraints that describe the feasible set. Following the

transformation described in Section 8.3.1 in this volume, the problem can be transformed into a linear programming problem, which can be written as

$$
\begin{aligned}
\text{P: Min } z &= y_1 + y_2 \\
\text{s.t.} \quad y_1 &\geq 7 - x_1 \\
y_1 &\geq x_1 - 7 \\
y_2 &\geq 4 - x_2 \\
y_2 &\geq x_2 - 4 \\
x_2 &\leq 6 \\
x_1 + x_2 &\leq 8 \\
2x_1 + x_2 &\leq 11 \\
x_1 &\leq 5 \\
x_1, \; x_2 &\geq 0.
\end{aligned}
$$

The optimal solution of this problem is $\mathbf{x} = (\bar{x}_1, \bar{x}_2) = (3\frac{1}{2}, 4)$ with an ℓ_1 distance of $3\frac{1}{2}$. Note that the optimal solution is not an extreme point of the original feasible set.

Employing the Chebyshev metric, i.e., using a minimax objective instead, i.e., Min z = max $\{y_1, y_2\}$, we can write the objective as Min z and add the constraints $z \geq y_1$ and $z \geq y_2$, and leave the remaining constraints unchanged. The problem then has a solution $\bar{\mathbf{x}} = (4\frac{2}{3}, 1\frac{2}{3})$ with a value of the objective function of $\bar{z} = 2\frac{1}{3}$.

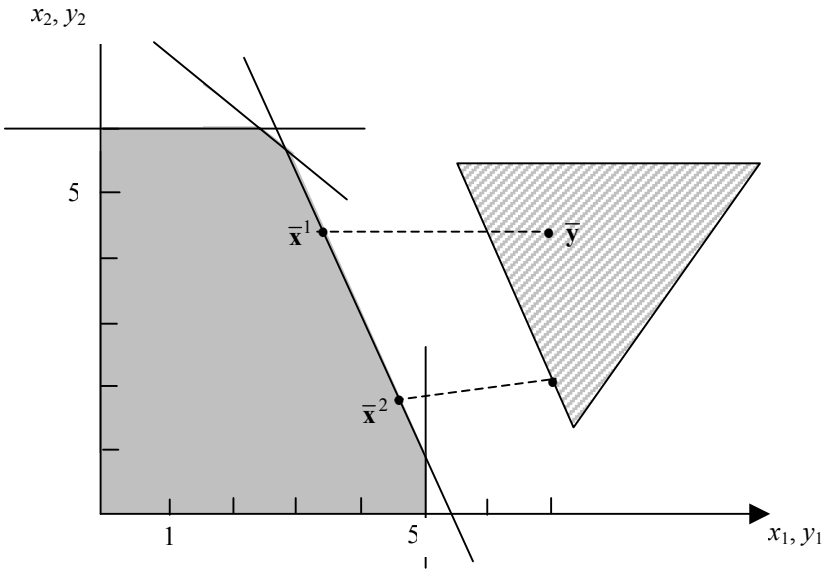


Figure 9.8

An extension of this concept would include not a single ideal point (which may be very difficult to specify by the decision maker), but an "*indifference region*" of points, which are all deemed desirable by the decision maker. In order to formulate the problem, we define the variables x_1 and x_2 as usual in the feasible set of the above example, and additional variables y_1 and y_2 that determine a point in the indifference region, which are restricted by the boundaries of that region. Let the constraints that define the indifference region be

$$\begin{aligned} y_2 &\leq 5 \\ 3y_1 + y_2 &\geq 23 \\ 3y_1 - 2y_2 &\leq 17. \end{aligned}$$

The situation is shown in Figure 9.8 with the shaded area indicating the feasible set and the crosshatched area showing the indifference region.

In order to determine the shortest possible ℓ_1 distance between a feasible point (x_1, x_2) and a point in the indifference region (y_1, y_2), we minimize $|x_1 - y_1| + |x_2 - y_2|$. This can again be accomplished by defining two new variables z_1 and z_2 and formulating the problem as

$$\text{P: Min } z = z_1 + z_2$$

$$\begin{aligned} z_1 &\geq x_1 - y_1 \\ z_1 &\geq y_1 - x_1 \\ z_2 &\geq x_2 - y_2 \\ z_2 &\geq y_2 - x_2 \end{aligned}$$

$$\begin{aligned} y_2 &\leq 5 \\ 3y_1 + y_2 &\geq 23 \\ 3y_1 - 2y_2 &\leq 17 \end{aligned}$$

$$\begin{aligned} x_2 &\leq 6 \\ x_1 + x_2 &\leq 8 \\ 2x_1 + x_2 &\leq 11 \\ x_1 &\leq 5 \\ x_1, \quad x_2 &\geq 0 \end{aligned}$$

The optimal solution has $\mathbf{x} = (\bar{x}_1, \bar{x}_2) = (4½, 2)$ and $\mathbf{y} = (\bar{y}_1, \bar{y}_2) = (7, 2)$ with a distance of $\bar{z} = 2½$.

9.3.2 Fuzzy Programming

It is apparent that the information provided to an analyst or decision maker is not always correct or clear. In particular, there are two types of lack of precision:

ambiguity applies if somebody is not sure about a number ("the demand for our product is probably about 500 this month") and *vagueness*, which indicates that somebody is not able to clearly specify an achievement ("our profit this quarter should be significantly above $100,000"). Both types of imprecision can be dealt with by fuzzy sets. In this section, we deal with vagueness.

Fuzzy sets were introduced by Bellman and Zadeh (1970). Much of the early work on their use was done by Zimmermann (1976, 1978). This an active area that even has its own journal "Fuzzy Sets and Systems." Good surveys can be found in Zadeh (1979) and Inuiguchi and Ramík (2000). Here, we first introduce the concept of fuzzy sets and provide a basic model that allows us to incorporate fuzziness in a linear optimization model.

Our point of departure is the standard set theory, in which given elements x_1, x_2, …, x_n are included in the universal set U. Suppose that there are sets A, B, … $\subset U$, and each arbitrary element $x_j \in A$ or $x_j \notin A$ for a set A. This concept is usable if everything is well defined, e.g., if the set A is defined as "the set of corporations that are traded at the NYSE and that reported profits of strictly more than $10 million in 2006." Each company then either is an element of that set or it is not and there is no gray zone. Sets that are defined in a clear and concise way are termed *crisp*. However, we can "fuzzify" the statement simply by replacing "…profits of strictly more then $10 million…" by "…profits of significantly more than $10 million… ." The term "significantly" is subjective and the set A is no longer crisp but *fuzzy*.

Given a fuzzy set A, we can define a so-called *fuzzy membership function* $\mu_A(x)$, which expresses the degree to which each individual element $x \in A$ belongs to the set A. The fuzzy membership function assumes a value of zero, if the element in question is very far from or most certainly not a member of the set, and it equals one if the element is definitely part of the set. The case of the usual crisp sets is a special case of fuzzy sets with a membership function that equals either zero or one. In the above example, let the elements x_1, x_2, …, x_7 denote the profits of six companies (in millions), such that $x_1 = 9$, $x_2 = 10$, $x_3 = 13$, $x_4 = 18$, $x_5 = 20$, and $x_6 = 30$. Defining now a fuzzy membership function as

$$\mu_A(x) = \begin{cases} 0, \text{ if } x \le 10 \\ \dfrac{x-10}{20-10}, \text{ if } 10 < x < 20 \\ 1, \text{ if } x \ge 20 \end{cases}$$

The values of the membership function are then $\mu_A(x_j) = 0$, 0, .3, .8, 1, and 1. The first two elements are not part of the set as they do not satisfy the specification of a minimum of $10 million in profits, the companies 3 and 4 are associated with

gradually increasing degrees of the membership function, while companies 5 and 6 are both thought of as having profits that are significantly higher than \$10 million, so both of them are full members of the set A.

Notice the difference between the fuzzy membership function and probabilities: while the membership function measures the degree to which an element satisfies an ill-defined constraint or objective, a probability would measure the likelihood that an element satisfies a well-defined constraint or objective.

We are now able to define some basic operations on fuzzy sets. Let x_j, j=1, …, n denote elements. We then can write

Definition 9.3: The *intersection* of two fuzzy sets A and B with membership functions $\mu_A(x)$ and $\mu_B(x)$ is defined as $C = A \cap B$ with $\mu_C(x_j) = \min \{\mu_A(x_j), \mu_B(x_j)\}$ $\forall\, j$.

Similarly we can define

Definition 9.4: The *union* of two fuzzy sets A and B with membership functions $\mu_A(x)$ and $\mu_B(x)$ is defined as $D = A \cup B$ with $\mu_D(x_j) = \max \{\mu_A(x_j), \mu_B(x_j)\}$ $\forall\, j$.

As a simple illustration, consider the following numerical

Example: There are four elements x_1, x_2, x_3, and x_4, whose membership functions are $\mu_A(x_j) = .8, .3, 0,$ and 0 as well as $\mu_B(x_j) = 0, .4, .2, .7$. The intersection $C = A \cap B$ has memberships $\mu_C(x_j) = 0, .3, 0, 0$, while the intersection $D = A \cup B$ has membership function $\mu_D(x_j) = .8, .4, .2,$ and .7.

Again, it is easy to see that these definitions have the equivalent operations for crisp sets as a special case. Consider three elements x, y, and z, such that $x, y \in A$ and $y, z \in B$, we have membership functions $\mu_A =$ 1, 1, 0 and $\mu_B = 0, 1, 1$, so that the $A \cap B$ has values of the membership function of 0, 1, 0, indicating that only y is a member of the intersection, while the membership values of the union are 1, 1, 1, which coincides with the usual result.

Consider now the objective function in an optimization problem and how it can be remodeled by using fuzzy sets. Suppose that a firm wants to maximize some objective function and assume that the decision maker is able to provide some upper and lower bounds for the measure of interest. Assume that the measure of interest is profit and assume that the decision maker has specified a lowest acceptable profit of \$5 million and a desirable target value of \$10 million We can then define a set A that includes all solutions whose profit significantly exceeds \$5 million. The membership function $\mu_A(x)$ would then be zero for all solutions with profits of less than \$5 million, it would assume a value of one for all solutions with profits of \$10 or more, and it would have some value between zero and one for all solutions whose associated profits are between \$5 and \$10 million.

Suppose now that the decision maker has a second objective, called customer satisfaction. This qualitative criterion has been measured on a five-point Likert scale with "1" denoting an unhappy customer and "5" symbolizing a customer who is completely satisfied. Suppose our decision maker would accept a neutral customer who is satisfied with a rating of "4," so that we can define a set B that includes all solutions that have a satisfaction rating substantially above neutral. The membership function then has all solutions with satisfaction rating below "3" at the zero level, all solutions x with a rating of "4" or higher will have $\mu_B(x) = 1$, and the solutions with satisfaction ratings between "3" and "4" have values between zero and one.

Considering now both objectives, the decision maker would like to have profits "significantly higher than \$5 million," i.e., membership in the set A, as well as customer satisfaction ratings, "well above neutral," i.e., membership in the set B. In order to satisfy both criteria, we have to find solutions in the intersection $C = A \cap B$. Following Definition 9.3, membership in the set C is defined by the membership function $\mu_C(x) = \min \{\mu_A(x), \mu_B(x)\}$, which, since both measures are utilities (rather than disutilities), is to be maximized. This leads to the following reformulation. Consider a standard vector optimization problem

$$\begin{aligned} \text{P: "Max"} \; & z_1 = \mathbf{c}^1\mathbf{x} \\ & z_2 = \mathbf{c}^2\mathbf{x} \\ & \dots \\ & z_r = \mathbf{c}^r\mathbf{x} \\ \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}, \end{aligned}$$

in which the decision maker has specified lower bounds L_k and upper bounds U_k on the values of the objective functions $k = 1, \ldots, r$ as well as defined the membership function that will assume values of zero for objective values below L_k, one for objective values above U_k, and values between zero and one for objective values between those bounds. One possibility (but by no means the only one) is to employ a piecewise linear membership function

$$\mu_k(x) = \begin{cases} 0, \text{ if } \mathbf{c}^k\mathbf{x} \leq L_k \\ \dfrac{\mathbf{c}^k\mathbf{x} - L_k}{U_k - L_k}, \text{ if } \mathbf{c}^k\mathbf{x} \in]L_k, U_k[\\ 1, \textit{if } \mathbf{c}^k\mathbf{x} \geq U_k \end{cases}$$

The problem can then be formulated as

$$\begin{aligned} \text{P':}\ \text{Max } z' &= \min_k \{\mu_k(\mathbf{x})\} \\ \text{s.t.}\quad & \mathbf{Ax} \le \mathbf{b} \\ & \mathbf{x} \ge \mathbf{0}. \end{aligned}$$

Using the standard reformulation described in Section 8.3.3 of this volume, we can rewrite this problem as

$$\begin{aligned} &\text{P":}\ \text{Max } z" \\ &\quad \text{s.t.}\ \ z" \le \mu_k(\mathbf{x})\ \forall\ k \\ &\quad \text{s.t. } \mathbf{Ax} \le \mathbf{b} \\ &\qquad \mathbf{x} \ge \mathbf{0}. \end{aligned}$$

Since $\mu_k(\mathbf{x})$ are piecewise linear functions, we may proceed to solve problem P" as follows. First, we introduce the constraints $\mathbf{c}^k\mathbf{x} \ge L_k\ \forall\ k$ and $\mathbf{c}^k\mathbf{x} \le U_k\ \forall\ k$ and replace the constraints $z" \le \mu_k(\mathbf{x})$ by $z" \le \dfrac{\mathbf{c}^k\mathbf{x} - L_k}{U_k - L_k}$. If the problem has no feasible solution because at least one of the lower bounds cannot be achieved, then the lowest achievable value of any of the membership functions equals zero. Objectives that have reached the upper bound can have their upper bound removed and their value $\mu_k(\mathbf{x})$ replaced by one. If the upper bounds are chosen as the maximal achievements of an objective as all other objectives are disregarded, then these upper bound constraints do not have to be included in the reformulation.

In order to illustrate the above approach, consider again the problem that was solved for vector optimization problems in Section 9.1. For convenience, we restate the problem here.

Example: The bi-objective linear optimization problem under consideration is

$$\begin{aligned} &\text{P: ``Max''}\ z_1 = -2x_1 + 3x_2 \\ &\quad \text{``Max''}\ z_2 = 3x_1 - x_2 \\ &\quad \text{s.t.} \qquad -x_1 + x_2 \le 2 && (1) \\ &\qquad\qquad x_1 + 2x_2 \le 6 && (2) \\ &\qquad\qquad x_1 \le 4 && (3) \\ &\qquad\qquad x_1 + x_2 \ge 1 && (4) \\ &\qquad\qquad x_1,\ \ x_2 \ge 0. \end{aligned}$$

Solving two simple linear programming problems results in the ideal point $(\bar{z}_1, \bar{z}_2) = (U_1, U_2) = (6⅔, 12)$. The least acceptable (nadir) point is set somewhat arbitrarily to $(L_1, L_2) = (⅔, 2)$. Assuming that both objective values are between their lower and upper bounds, the objective function of the problem can then be reformulated as

$$\text{P: Max } z = \min\left\{\frac{(-2x_1+3x_2)-\frac{2}{3}}{6}, \frac{(3x_1-x_2)-2}{10}\right\}$$
$$= \min\left\{(-1/3x_1+1/2x_2-1/9), (.3x_1-.1x_2-.2)\right\}.$$

The problem can then be written as

P': Max z

s.t. $z \le -\frac{1}{3}x_1 + \frac{1}{2}x_2 - 1/9$
$z \le .3x_1 - .1x_2 - .2$

$$\text{s.t. } -x_1 + x_2 \le 2 \quad (1)$$
$$x_1 + 2x_2 \le 6 \quad (2)$$
$$x_1 \le 4 \quad (3)$$
$$x_1 + x_2 \ge 1 \quad (4)$$
$$x_1, \quad x_2 \ge 0.$$

The optimal solution to problem P' is $\bar{x}_1 = 2.0238$, $\bar{x}_2 = 1.9881$ so that $\bar{z} = 0.2083$. It is apparent that constraint (2) is binding at optimum and the optimal solution is on the nondominated frontier shown in Figure 9.5a.

There is a variety of extensions of this basic approach. One of them is to include nonlinear fuzzy membership functions, which then have to be dealt with by techniques from nonlinear optimization, while the inclusion of fuzzy parameters leads to *possibilistic programming*. An interesting paper in this context is by Vasant *et al*. (2005). However, such extensions are beyond the scope of this book.

Other extensions of fuzziness are derived from the theory of *rough sets* as introduced by Pawlak (1991). Their use in linear programming, or optimization in general has, however, not been established.

9.3.3 Goal Programming

Another way to address optimization problems in case the decision maker has provided some guidelines concerning values of the objective functions is *goal programming*. The essence of goal programming is the introduction of *aspiration levels* or *target values* t_k, k=1, …, r with the proviso that the solution should reach the target values, *if possible*. In the sense that they provide externally determined reference points, goal programming problems are similar to reference point methods. The formulation of problems with target values typically blurs the distinction between objectives and constraints, since objectives are formulated as goal constraints, so that the deviation of the actual achievement is measured in terms of deviation from the goal and this deviation is then minimized.

The earliest descriptions of goal programming are due to Charnes *et al*. (1955) and Charnes and Cooper (1961), where in the latter reference, the authors also introduce the term "goal programming." Ijiri (1965) uses a preemptive priority structure for the individual goals. Ignizio (1982) summarizes his work in the 1960s and 1970s, which includes may applications of goal programming. Other milestones are the books by Lee (1972) and Schniederjans (1984). Newer contributions are the books by Ignizio and Cavalier (1994), Schniederjans (1995), and the edited collections by Trzaskalik and Michnik (2002) and Tanino *et al*. (2003).
The underlying concept in the goal programming approach is the concept of *satisficing* introduced by Simon (1957). Simply speaking, the main idea of satisficing (= satisfying + sufficing) is that decision makers do not necessarily maximize their utility, but instead, are satisfied when a predetermined target value or aspiration level has been achieved.

The modeling procedure progresses as follows. The first step is to decide which of the constraints are "hard," i.e., must be satisfied under all circumstances. Those constraints will be dealt with as usual. Consider now the *goal constraints*, i.e., those constraints that should be satisfied whenever possible, but if they are not, we would be satisfied if the solution would be as close as possible to the prescribed target.

Formally, consider a goal constraint of the type $\mathbf{a}_{\ell\bullet}\mathbf{x}\ R\ t_\ell$. where the left-hand side, as usual, denotes the achievement by the present solution, whereas the right-hand side denotes a target value or aspiration level t_ℓ with which the present achievement will be compared. In contrast to regular constraints, goal constraints do allow deviations of the actual achievement from the target value. For that purpose, we define *deviational variables* d_ℓ^- that measure the *underachievement* of the target and as such act as slack variables, as well as deviational variables d_ℓ^+ that measure *overachievements*, which can be interpreted similar to excess variables. The objective function will then be composed of a weighted sum of over- and underachievements of the individual measures.

Given the three admissible relations in linear programming, we can distinguish between three situations shown in Table 9.4.

Table 9.4

Desired situation	Formulation of goal constraint	Contribution to the objective function
$\mathbf{a}_{k\bullet}\mathbf{x} \le t_\ell$	$\mathbf{a}_{\ell\bullet}\mathbf{x} + d_\ell^- - d_\ell^+ = t_\ell$ $d_\ell^-, d_\ell^+ \ge 0$	Min d_ℓ^+
$\mathbf{a}_{k\bullet}\mathbf{x} = t_\ell$		Min $d_\ell^- + d_\ell^+$
$\mathbf{a}_{k\bullet}\mathbf{x} \ge t_\ell$		Min d_ℓ^-

As an illustration, suppose that 100 resource units are available and only 80 have been used in a particular solution. Since both deviational variables are constrained to be nonnegative, d_ℓ^-=20 and $d_\ell^+ = 0$ will result. This indicates that twenty units less than available are used in the solution. Similarly, if we were to have used 130 units in some solution, the deviational variables would assume the values $d_\ell^- = 0$ and $d_\ell^+ = 30$, indicating that we overuse the resources by 30 units. Note that had this constraint been formulated as a regular constraint, such an overuse would not have been allowed. In contrast, a goal constraint will allow overuse, but minimizing d_ℓ^+ in the objective function will attempt to keep such overuse as small as possible. In contrast to regular constraints that limit achievements, goal constraints will asked to respect an achievement, *if possible*. This softening of the constraints allows deviations that carry a penalty with them, but are nonetheless possible.

The fact that we use over- and underachievement variables in the same constraint may appear to be a problem. However, it is not possible that both variables are positive at the same time. The reason is that the columns of the two variables are identical except for the sign, i.e., they are linearly dependent, so that they cannot be included in the same basic matrix which, by definition, has full rank.

One serious problem arises in the objective function, though. The objective function in goal programming is constructed as the weighted sum of deviations from the target values. This means that one single objective will include deviational variables that are defined in terms of dollars, others in terms of manpower, risk, calories, tons of steel, or whatever the application may call for. Clearly, including all of these variables with different units together in one objective raises the issue of *commensurability*. One easy (but potentially not satisfactory) way out of this is to define weights, whose function is not only to express the different degrees of importance of the over- and underachievements, but also to provide a conversion of the units used in the goal constraints.

As described above, goal programming can be classified as a reference point method. As opposed to the reference point method described earlier, goal programming employs target values that are reachable and the solution can deviate from both sides.

One other feature that may be present in goal programming problems is a lexicographic structure of the overall objective. In particular, it is possible to have a finite number of individual objectives, each assumed to be infinitely more important as the next lower-ranking objective. On each such level, finite weights are used to ensure that deviational variables from different goal constraints are commensurable and their relative importance is expressed properly.

The notion that classes of constraints are infinitely more important than others—while standard in lexicographic programming problems—appears nevertheless contrived. Such a concept is, however, not at all new to users of linear programming. Consider a standard linear programming problem with the usual constraints and an objective function. Here it is infinitely more important to satisfy the constraints than optimizing the objective function, as without a feasible solution, optimization is meaningless. In other words, the preemptive priority structure applies there as well. In goal programming, this structure is generalized to apply to different levels of objective functions.

In order to formally define a goal programming problem, we have to introduce some definitions. The regular constraints in a goal programming problem are supposed to include slack and excess variables as usual (which here, for convenience, are denoted as x_j similar to the decision variables). Then artificial variables are added to those constraints that are originally of the type ≥ and =, as usual. The numbers of the constraints that involve artificial variables are collected in the set I, so that artificial variables A_i exist for all constraints in the set I.

All goal constraints are written with deviational variables as shown in Table 9.4. The deviational variables are then collected into clusters depending on their importance. The highest-ranking and most important deviations are collected in cluster 1, cluster 2 includes the next class of importance, and so forth. The preemptive priority levels are then $P_0 >>> P_1 >>> P_2 >>> \ldots >>> P_r$. The model can then formally be written as

$$\text{P: Min } z = P_0 \sum_{i \in I} A_i + \sum_{k=1}^{r} P_k \left[\sum_{\ell=1}^{s} w_{\ell k}^{-} d_{\ell}^{-} + \sum_{\ell=1}^{s} w_{\ell k}^{+} d_{\ell}^{+} \right]$$

$$\text{s.t.} \quad \sum_{j=1}^{n} a_{ij} x_j + A_i = b_i \quad \forall \; i \in I$$

$$\sum_{j=1}^{n} a_{ij} x_j + d_{\ell}^{-} - d_{\ell}^{+} = t_{\ell} \quad \forall \; \ell = 1,\ldots s$$

$$x_j, A_i, d_{\ell}^{-}, d_{\ell}^{+} \geq 0 \quad \forall \; j, i, \ell$$

If the model also includes a regular maximization or minimization objective function, it can also be incorporated in this structure. Assume that the objective were to maximize the function **cx**, we can then define an (unattainable) target value z for it, write the goal constraint $\mathbf{cx} + d_h^{-} - d_h^{+} = z$ and minimize the underachievement d_h^{-}. If the target value z is chosen sufficiently large, this is equivalent to maximizing the function **cx**.

In order to illustrate the modeling with goal constraints, consider the following

Example: An investor has \$100,000 that can be invested in three alternatives, *viz.*, bonds, stocks, and the family business. The respective returns are 6%, 8% and 4%. In addition to the money available, it is possible to borrow money from the bank at 9%. The hard constraints and the softer goal constraints are listed below.

Absolute constraints:

(1) Do not borrow more than \$20,000 from the bank.
(2) Do not overspend the total budget.
(3) The amount of money invested in the family business has to exceed the total amount of money invested elsewhere by at least \$10,000.

Priority level 1:

(4) Due to risk aversion, at most 10% of the total amount of money invested can be allocated to stocks, if possible
(5) Security reasons require an investment of \$40,000 or more in bonds, if possible
(6) The family asks for \$30,000 or more for their business, if possible.

The decision maker assumes that the three goals (4), (5), and (6) have relative weights of 2, 3, and 7, respectively.

Priority level 2:

(7) Invest exactly \$100,000, if possible
(8) Maximize the return on the total investment.

The decision maker specifies that both objectives on this level have equal weight. In addition, the objectives on priority level 1 are assumed to be infinitely more important than those on priority level 2.

We first define variables x_1, x_2, and x_3 as the dollar amounts invested in the three alternatives and x_4 as the amount borrowed from the bank. The constraints can then be formulated as

$$\begin{aligned} x_4 &\leq 20{,}000 && (1') \\ x_1 + x_2 + x_3 &\leq 100{,}000 + x_4 && (2') \\ x_3 &\geq x_1 + x_2 + 10{,}000 && (3') \end{aligned}$$

Including slack variables x_5 and x_6 as well as an excess variable x_7 and an artificial variable A_1, the constraints can be written as

$$\begin{aligned} x_4 + x_5 &= 20{,}000 && (1) \\ x_1 + x_2 + x_3 + x_6 &= 100{,}000 + x_4 && (2) \\ x_3 - x_7 + A_1 &= x_1 + x_2 + 10{,}000 && (3) \end{aligned}$$

If the goal constraints on the first priority level were to be formulated as regular (i.e., hard) constraints, we would write

$$x_2 \le .1(x_1 + x_2 + x_3) \quad (4')$$
$$x_1 \ge 40{,}000 \quad (5')$$
$$x_3 \ge 30{,}000 \quad (6')$$

Similarly, formulating the goal constraints on the second priority level as hard constraints, we would like to obtain (if possible)

$$x_1 + x_2 + x_3 = 100{,}000 \quad (7')$$
$$\text{Max } z = .06x_1 + .08x_2 + .04x_3 - .09x_4 \quad (8')$$

Using now deviational variables and the transformation shown in Table 9.4, we can write the goal constraints that derive from the relations (4'), (5'), and (6') as

$$x_2 + d_4^- - d_4^+ = .1(x_1 + x_2 + x_3) \quad (4)$$
$$x_1 + d_5^- - d_5^+ = 40{,}000 \quad (5)$$
$$x_3 + d_6^- - d_6^+ = 30{,}000 \quad (6)$$

with objective function contributions of Min d_4^+, Min d_5^-, and Min d_6^-, respectively.

The goal constraints on the second level are rewritten as

$$x_1 + x_2 + x_3 + d_7^- - d_7^+ = 100{,}000 \quad (7)$$
$$.06x_1 + .08x_2 + .04x_3 - .09x_4 + d_8^- - d_8^+ = z \quad (8)$$

where we can set $z := 10{,}000$ as an unattainable level of achievement. Furthermore, the objective function contributions are Min $d_7^- + d_7^+$ and Min d_8^-, respectively.

The highest priority level includes the artificial variables of the regular constraints. It can be written as

$$\text{Min } A_1.$$

The next priority level includes weighted deviations defined in the goal constraints

$$\text{Min } 2d_4^+ + 3d_5^- + 7d_6^-,$$

and the next and lowest priority level includes

$$\text{Min } d_7^- + d_7^+ + d_8^-.$$

The overall objective is then

$$\text{Min } P_0 A_1 + P_1(2d_4^+ + 3d_5^- + 7d_6^-) + P_2(d_7^- + d_7^+ + d_8^-).$$

There are different ways to solve goal programming problems. The simplest way (and one that uses existing linear programming software) is the *sequential goal programming* approach. On level k, this technique proceeds as follows. Assume that the objective function of the original problem is written as

$$\text{Min } z = \sum_k P_k z_k ,$$

where z_k is the appropriate function of deviational variables. Furthermore, assume that the values on the levels $\ell = 1, \ldots, k$ have already been determined as $\bar{z}_1, \bar{z}_2 ..., \bar{z}_{k-1}$. On level k, the problem under consideration can then be written as

$$\begin{aligned} \text{P: Min } & z_k \\ \text{s.t. } & z_\ell \le \bar{z}_\ell , \ell = 1, \ldots, k–1 \\ & \mathbf{Ax} \le \mathbf{b} \\ & \mathbf{x} \ge \mathbf{0}. \end{aligned}$$

In other words, in iteration k the method will add constraints that require the higher levels to be restricted to the values they achieved when those levels were optimized. Starting with the highest-ranking part of the objective, this process is performed for all levels sequentially—hence the name—until the lowest level has been optimized.

As an illustration, consider the investment example above. On Level 0, the objective minimizes A_1. There is an infinite number of feasible solution that have A_1 at the zero level. Deleting the artificial variable keeps it at the zero level indefinitely. Consider now Level 1. The objective is Min $z_1 = 2d_4^+ + 3d_5^- + 7d_6^-$ which is optimized under consideration of the constraints (1) – (8), leading to the solution $\bar{x}_1 = 40{,}000$, $\bar{x}_2 = 0$, $\bar{x}_3 = 50{,}000$, and $\bar{x}_4 = 0$ with $\bar{z}_1 = 0$.

On Level 2, we employ the objective Min $z_2 = d_7^- + d_7^+ + d_8^-$ coupled with the constraints (1) – (8) and the additional constraint that requires the achievement on the higher-ranking Level 1 to be at least as good as in the previous step, i.e., $2d_4^+ + 3d_5^- + 7d_6^- \le 0$. The optimization results in $\bar{x}_1 = 40{,}000$, $\bar{x}_2 = 5{,}000$, $\bar{x}_3 = 55{,}000$, and $\bar{x}_4 = 0$ with the partial objective achievement of $\bar{z}_2 = 95{,}000$.

Other solution approaches to goal programming problems exist. For instance, there is a multiphase optimization method, which uses some properties to reduce the size of the problem and it also avoids having to re-solve the problem for each

level. However, given the computational power available today and the relatively small number of levels in practical problems, it does not appear worthwhile to spend much effort in devising other solution techniques.

It may be very tempting to avoid resolving the problem for each level separately by using finite weights $P_1 >> P_2 >> \ldots >> P_r$ and then solving the resulting linear programming problem once. However, this reformulation may not result in the correct solution.

Example: A single-level goal programming problem is given as

$$\text{P: Min } z = P_1(d_1^- + d_1^+) + P_2 d_2^-$$

$$\begin{aligned} \text{s.t.} \quad & x_1 + x_2 \le 2 \\ & x_1 + d_1^- - d_1^+ = 3 \\ & 1{,}000x_2 + d_2^- - d_2^+ = 5{,}000 \\ & x_1, x_2, d_1^-, d_1^+, d_2^-, d_2^+ \ge 0. \end{aligned}$$

The optimal solution of this goal programming problem is $\bar{x}_1 = 2$, $\bar{x}_2 = 0$, $d_1^- = 1$, $d_1^+ = d_2^+ = 0$, and $d_2^- = 5{,}000$. If we were to set $P_1 := 100$ and $P_2 := 1$ in order to mimic the large difference between the importance of the two levels, and then aggregate the two levels of the objective, we obtain the solution $\bar{x}_1 = 0$, $\bar{x}_2 = 2$, $\bar{d}_1^- = 3$, $\bar{d}_2^- = 3{,}000$, and $\bar{d}_2^+ = \bar{d}_2^+ = 0$, obviously a very different solution.

We will conclude this section by providing another possible application of goal programming. Suppose we have a system of simultaneous linear equations $\mathbf{Ax} = \mathbf{b}$ that does not have a feasible solution, i.e., some of the statements in the equations are contradictory. Since there is no feasible solution $\mathbf{x} \in \mathbb{R}$, we may want to determine a solution that is as close as possible to the feasible set described by the equations. Assume that we define proximity, i.e., the degree by which a solution deviates from the feasible set by the sum of deviations from feasibility, we can then solve the single-level goal programming problem

$$\text{P: Min } z = \sum_i (d_i^- + d_i^+)$$

$$\begin{aligned} \text{s.t.} \quad & \sum_j a_{ij} x_j + d_i^- - d_i^+ = b_i \ \forall\, i \\ & x_j \in \mathbb{R} \ \forall\, j;\ d_i^-, d_i^+ \ge 0 \ \forall\, i. \end{aligned}$$

In order to explain the procedure, consider the following

Example: Consider the system of simultaneous linear equations

$$\begin{aligned} x_1 + 3x_2 + 7x_3 &= 120 \\ -3x_1 + x_2 + 2x_3 &= 60 \\ -x_1 - 3x_2 + 5x_3 &= 60. \end{aligned}$$

It is easy to verify that the system has no solution. We now attach deviational variables to the equations and minimize their sum in the system

$$\begin{aligned} \text{P}'\text{: Min } z = {} & d_1^- + d_1^+ + d_2^- + d_2^+ + d_3^- + d_3^+ \\ \text{s.t.} \quad & x_1 + 3x_2 + 7x_3 + d_1^- - d_1^+ = 120 \\ & -3x_1 + x_2 + 2x_3 + d_2^- - d_2^+ = 60 \\ & -x_1 - 3x_2 + 5x_3 + d_3^- - d_3^+ = 60. \end{aligned}$$

This problem has a solution $\bar{x}_1 = 0$, $\bar{x}_2 = 5$, and $\bar{x}_3 = 15$ with a total deviation of $\bar{z} = 25$. If we were to define the deviation of a solution from feasibility not as the sum of deviations but equal to the largest single deviation (for the appropriate reformulation, see Section 8.3.3 in this volume), we obtain the solution $\bar{x}_1 = 0$, $\bar{x}_2 = 40$, and $\bar{x}_3 = 2.8571$ with the largest deviation in the each of the three constraints of $\bar{z} = 14.2857$.

Many extensions and practical approaches have been suggested to the basic goal programming model. Early interactive techniques were discussed by Dyer (1972), more recent interactive approaches are described by Reeves and Hedin (1993) and Zykina (2004). Jones and Tamiz (2002) provide a survey of the field in the years up to 2000. Two issues of the journal INFOR in 2004 have also been devoted to a variety of aspects of goal programming.

9.4 Bilevel Programming

Bilevel problems have a long history in economics. Their structure includes two levels on which decisions are made. On the upper level, there is the *leader*, while the *follower* occupies the lower level The first to describe situations of this nature was the economist von Stackelberg (1934) and the problems were introduced into the operations research field by Bracken and McGill (1973). A comprehensive account of bilevel programming is provided by Dempe (2002) and up-to-date surveys have been provided by Vicente and Calamai (1994), Vicente (2001), Colson *et al*. (2005), and Fliege and Vicente (2006).

The basic idea in bilevel programs is that the (market) leader is able and willing to make his decision first, not knowing what the followers will do and how they will react. Once the leader has made his decision, the follower(s) will take the leader's decision as given and react in a way that optimizes their own objective. The

marketing concept of the "first mover" is closely related. Strictly speaking, the concept belongs under the umbrella of sequential games. Some of the issues related to leader – follower games deals with the question whether or not the leader actually has an advantage over the follower.

It is also noteworthy that in order to become a leader, a firm or decision maker has to have the ability and an incentive to do so. As an example, consider the pharmaceutical industry. A firm will only (be able to) develop a new drug, if it has sufficient resources, and, given that, it will only proceed with the development if there is a sufficiently great incentive to do so, which is usually regulated by the patenting laws. These laws will also determine whether or not there is an advantage to the market leader.

Notice the asymmetry in leader – follower games: while the leader faces the potentially uncertain reaction of the followers (requiring him to guard against the follower's possible decisions), the follower faces a fully deterministic situation in which he only has to take the leader's decision into account, which has already been made by the time the follower has to make his decision. Given this, it is apparent that from a computational point of view, the follower's problem is considerably easier to solve.

Bilevel problems are usually written in the form Max $f(\mathbf{y}, \mathbf{x})$, s.t. $g(\mathbf{y}, \mathbf{x}) \leq \mathbf{0}$, where $\mathbf{y}$ is a vector of variables that is under the jurisdiction of the (von Stackelberg) leader, while $\mathbf{x}$ includes the variables that are within the jurisdiction of the follower. This scenario lends itself easily to competitive situations, but not as obviously to general multiobjective scenarios. One possibility is the following. A large retail company sells shoes that it purchases from a number of suppliers. Acting as a leader, it will want to set price, quality, and quantity. However, the smaller supplier has the possibility to negotiate some of the conditions as it may otherwise deal with another retailer instead. Note that this scenario only applies if there are multiple retailers that work independently and there is sufficient demand for the supplier's product.

The general idea of solving bilevel problems is to first solve the follower's problem. In order to do so, the follower will take the leader's decision as given and derives an optimal response to each of the leader's possible decisions. This reaction function is then used by the leader in his optimization. As long as the follower's reaction function can be expressed in a closed form, the leader's problem is also easy. However, this is almost always not the case. Typically, the reaction function is the solution of a (linear) programming problem, which then becomes difficult to incorporate in the bilevel scenario.

In order to convey the basic ideas, consider the following illustration. We have chosen to use a small, albeit nonlinear, problem that has an easy solution and a meaningful interpretation.

Example: An employer and an employee are planning their work schedules. The employer is the leader as he will offer a certain wage. Given this wage, the employee, the follower, will plan how many hours he will offer for the wage set by the employer. To formalize, denote by w the wage rate, the employer's only variable, and let t_w and t_ℓ symbolize the work time and leisure time, respectively, decided upon by the employee. Suppose that the employee has a utility function $u(t_w, t_\ell) = wt_w + \alpha\sqrt{t_\ell}$, i.e., the employee's utility increases linearly with his wage and it increases at a decreasing rate with the leisure time he affords himself. The employee's work time is bounded by his desire to sleep for 8 hours, the unwillingness to work more than 14 hours, and a lower bound depending on the wage as $25/w$. The employee's utility maximizing problem can then be written as

$$\begin{aligned} P_f\text{: Max } & u(t_w, t_\ell) = wt_w + \alpha\sqrt{t_\ell} \\ \text{s.t.} \quad & t_w + t_\ell = 16 \\ & t_w \le 14 \\ & t_w \ge 25/w \\ & t_w, \quad t_\ell \ge 0. \end{aligned}$$

For any reasonable (nonnegative) wage rate, the nonnegativity constraints for the employee's two variables are redundant, and the first constraint can be used to eliminate one of the variables, leaving the problem

$$\begin{aligned} P_f\text{: Max } & u(t_w, t_\ell) = wt_w + \alpha\sqrt{16 - t_w} \\ \text{s.t.} \quad & t_w \in [25/w, 14]. \end{aligned}$$

On the other hand, the employer derives a (monetary) value of β out of each hour the employee works for him, while his costs equal the wage rate w. Consequently, the employer's problem is to maximize his monetary benefit (profit) π as

$$\begin{aligned} P_\ell\text{: Max } & \pi = (\beta - w)\, t_w \\ \text{s.t.} \quad & w \ge 0 \text{ and } t_w \text{ solves problem } P_f. \end{aligned}$$

In order to solve the problem, we temporarily take the wage rate w as given and consider the follower's problem. Taking the first derivative with respect to t_w and setting it to zero results in

$$\frac{du}{dt_w} = w - \tfrac{1}{2}\alpha(16 - t_w)^{-\frac{1}{2}} = 0 ,$$

resulting in

$$t_w = 16\text{-}\left(\frac{\alpha}{2w}\right)^2.$$

This function is the employer's reaction function. In other words, whatever wage the employer offers, the employee will use it in this relation to optimally plan his working time. Note that the relation does indeed specify a maximum as the second derivative is negative. For simplicity, assume that $\alpha = \sqrt{40}$.

Replacing t_w in the inequality constraints in P_f results in the conditions

$$w \le \frac{\alpha}{\sqrt{8}} = \sqrt{5} \approx 2.2361 \text{ and}$$
$$w \ge 1.8927, \text{ so that } w \in [1.8927, 2.2361].$$

Given the employee's reaction function, the employer's objective function is now

$$P_\ell\text{: Max } \pi = (\beta - w)\left(16 - \frac{\alpha^2}{4w^2}\right).$$

Letting now $\beta = 5$, we can take the derivative with respect to w, set it equal to zero, and solve. The result is a wage rate of $\overline{w} = 1.7291$. This is the leader's solution. Replacing this wage rate in the follower's reaction function leads to $\bar{t}_w = 12.6553$ hours of labor and $\bar{t}_\ell = 3.3447$ hours of leisure time.

The purpose of this example was to convey the basic strategy that is used to solve bilevel programming problems: In a recursive manner, the leader's variables are first considered parameters and the follower's problem is solved. The result is the follower's reaction function which is then replaced in the leader's problem. Finally, the leader uses this information to optimize his own model.

REFERENCES

Ackoff RL (1974) Redesigning the future: a systems approach to societal problems. Wiley, New York, NY

Ackoff RL (1978) The art of problem solving. Wiley, Inc., New York, NY

Ahuja RK, Magnanti TL, Orlin JB (1993) Network flows: theory, algorithms, and applications. Prentice-Hall, Englewood Cliffs, NJ

Anbil R, Gelman E, Patty B, Tanga R (1991) Recent advances in crew-pairing optimization at American Airlines. Interfaces 21: 62-74

Appa G, Smith C ((1973) On L_1 and Chebyshev estimation. Mathematical Programming 5: 73-87

Ballestero E, Romero C (1998) Multiple criteria decision making and its application to economic problems. Kluwer, Boston Dordrecht London

Barsow AR (1959) What is linear programming. Moscow: 90-101

Bazaraa MS, Sherali HD, Shetty CM (1993) Nonlinear programming: theory and algorithms. Wiley, New York, NY

Beale EML (1955) Cycling in the dual simplex algorithm. Naval Research Logistics Quarterly 2: 269-276

Bellman RE, Zadeh LA (1970) Decision-making in a fuzzy environment. Management Science 17: 141-164

Benayoun R, de Montgolfier J, Tergny J, Laritchev O (1971) Linear programming with multiple objective functions: step method (STEM). Mathematical Programming 1: 366-375

Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik 4: 238-252

Bhatti MA (2000) Practical optimization methods with Mathematica applications. Springer, New York, NY

Bland RG (1977) New finite pivoting rules for the simplex method. Mathematics of Operations Research 2: 103-107

Bracken J, McGill J (1973) Mathematical programs with optimization problems in the constraints. Operations Research 21: 37–44

Bradley SP, Hax AC, Magnanti TL (1977) Applied mathematical programming. Addison Wesley, Reading, MA

Bretscher O (1997) Linear algebra with applications. Prentice-Hall, Upper Saddle River, NJ

Brown GW, Koopmans TC (1951) Computational suggestions for maximizing a linear function subject to linear inequalities. In: Koopmans TC (ed) Chapter XXV in Activity analysis of production and allocation. Cowles Commission Monograph 13, Wiley, New York, NY, pp 377-380

Buchanan J, Gardiner L (2003) A comparison of two reference point methods in multiple objective mathematical programming. European Journal of Operational Research 149: 17-34

Bunch JR, Parlett BN (1971) Direct methods for solving symmetric indefinite systems of linear equations. SIAM Journal on Numerical Analysis 8: 639-655

Charnes A (1952) Optimality and degeneracy in linear programming. Econometrica 20: 160-170

Charnes A, Cooper WW (1961) Management models and industrial applications of linear programming. Wileys, New York, NY

Charnes A, Cooper WW, Ferguson RO (1955) Optimal estimation of executive compensation by linear programming. Management Science 1, 138-151

Charnes A, Cooper WW, Henderson A (1953) An introduction to linear programming. Wiley, New York

Charnes A, Cooper WW, Mellon B (1952) Blending aviation gasolines—a study in programming interdependent activities in an integrated oil company. Econometrica 20: 135-159

Charnes, A, Cooper WW, Rhodes E (1978) Measuring the efficiency of decision-making units. European Journal of Operational Research 2: 429-444

Charnes A, Klingman D (1971) The more-for-less paradox in the distribution model. Cahiers de centre d'études recherche operationnelle 13: 11-22

Charnes A, Lemke CE (1954) Computational theory of linear programming I: the bounded variables problem. ONR Research Memo No. 10, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA

Chen G, Huang X, Yang X (2005) Vector optimization. Lecture Notes in Economics & Mathematical Systems 541, Springer, Berlin

Cochrane JL, Zeleny M (eds) (1973) Multiple criteria decision making. University of South Carolina Press, Columbia, SC

Cohon, J. L. (1978): Multiobjective Programming Planning. In: Mathematics in Science and Engineering, Vol. 140, Academic Press, New York.

Collette Y, Siarry P (2003, 2nd edn.) Multiobjective optimization: principles and case studies. Springer, Berlin Heidelberg New York

Colson B, Marcotte P, Savard G (2005) Bilevel programming: a survey. 4OR 3: 87-107

Cook SA (1971) The complexity of theorem-proving procedures. Proceedings of the 3rd Annual ACM Symposium on the Theory of Computing, New York, pp. 151-158

Cooper L, Steinberg D (1970) Introduction to methods of optimization. W. B. Saunders Co., Philadelphia, PA

Dantzig GB (1951) Application of the simplex method to a transportation problem. In: Koopmans TC (ed), Activity analysis of production and allocation. Cowles Commission Monograph 13, Wiley, New York, pp 359-373

Dantzig GB (1954) Notes on linear programming, Parts VIII, IX, X, Upper bounds, Secondary constraints and block triangularity in linear programming. The Rand Corporation, Research Memorandum RM-1367, Oct 4, 1954. Published under the same title in Econometrica 23, 1955, 174-183

Dantzig GB (1963) Linear programming and extensions. Princeton University Press, Princeton, NJ

Dantzig GB (1982) Reminiscences about the origins of linear programming. In: Bachem A, Grötschel M, Korte B (eds). Mathematical programming. The State of the Art. Springer, Berlin Heidelberg New York, pp 78-86

Dantzig GB (1991) Linear programming. In: Lenstra JK, Rinnooy Kan AHG, Schrijver A (eds) History of mathematical programming: a collection of personal reminiscences. Elsevier Science, Amsterdam, pp 19-31

Dantzig GB, Ford LR, Fulkerson DR (1956) A primal-dual algorithm for linear programming. In: Kuhn HW, Tucker AW (eds) Linear inequalities and related systems. Annals of Mathematics Studies, Princeton University Press, Princeton, NJ, pp 171-181

Dantzig GB, Orchard-Hays W (1953) Notes on linear programming: part V – alternate algorithm for the revised simplex method using product form of the inverse. TM-1268, The RAND Corporation, Santa Monica, CA

Dantzig GB, Fulkerson DR, Johnson SM (1954) The solution of a large-scale traveling salesman problem. Operations Research 2: 393-410

Dantzig GB, Orden A (1953) Duality theorems, Rand report RM-1265. The Rand Corporation, Santa Monica, CA

Dantzig GB, Thapa MN (1997) Linear programming. I: Introduction. Springer, New York Berlin Heidelberg

Dantzig GB, Thapa MN (2003) Linear programming, 2: Theory and extensions. Springer, New York Berlin Heidelberg

Dantzig GB, Van Slyke RM (1967) Generalized upper bounded techniques for linear programming. Journal of Computer and Systems Sciences 1: 213-226

Dantzig GB, Wolfe P (1961) The decomposition algorithm for linear programs. Econometrica 29:767-778

Dantzig GB, Wolfe P (1960) Decomposition principle for linear programs. Operations Research 8:101-111

DeMarr R (1983) External pivoting in linear programming. Abstracts presented to the American Mathematical Society, issue 22, vol 4, no 1

Dempe S (2002) Foundations of bilevel programming, Kluwer, Boston, MA

Dikin I (1967) Iterative solution of problems of linear and quadratic programming. Soviet Mathematics Doklady 8: 674-675

Dinkelbach W (1969) Sensitivitätsanalysen und parametrische Programmierung. In: Ökonometrie und Unternehmensforschung XII, Springer, Berlin Heidelberg New York

Dwyer PS (1975) Transportation problems with some x_{ij} negative and transshipment problems. Naval Research Logistics Quarterly 22: 751-776

Dyer JC (1972) Interactive goal programming. Management Science 19: 62-70

Edmonds J (1965) Paths, trees, and flowers. Canadian Journal of Mathematics 17: 449-467

Egerváry J (1931) Matrixok kombinatorius tulajdonságairól (On combinatorial properties of matrices, in Hungarian). Mathematikai és Fizikai Lapok 38: 16-28

Ehrgott M (2005) Multicriteria optimization. Springer, Berlin Heidelberg New York

Eiselt HA; Laporte G (1987) Combinatorial optimization problems with soft and hard requirements. Journal of the Operational Research Society 38: 785-795

Eiselt HA, Pederzoli G, Sandblom C-L (1987) Continuous optimization. W. DeGruyter, Berlin

Eiselt HA, Sandblom C-L (1985) External pivoting in the simplex algorithm. Statistica Neerlandica 39: 327-341

Eiselt HA, Sandblom C-L (1990) Experiments with external pivoting. Computers & Operations Research 17: 325-332

Eiselt HA, Sandblom C-L (2000a) The bounce algorithm for mathematical programming. Mathematical Methods of Operations Research 52: 173-183

Eiselt HA, Sandblom C-L (eds and authors) (2000) Integer programming and network models. Springer, Berlin Heidelberg New York

Eiselt HA, Sandblom C-L (eds & authors) (2004) Decision analysis, location models, and scheduling problems. Springer, Berlin Heidelberg New York

Evans JP, Steuer RE (1973) A revised simplex method for linear multiple objective programming. Mathematical Programming 5: 54-72

Farrell JM (1957) The measurement of productive efficiency. Journal of the Royal Statistical Society 120: 253-281

Fiacco AV (1979) Barrier methods for nonlinear programming. In: Holzman AG (ed) Operations research methodology. Marcel Dekker, New York Basel, pp 377-440

Fiacco AV, McCormick GP (1968) Nonlinear programming: sequential unconstrained minimization techniques. Wiley, New York, NY

Figueira J, Greco S, Ehrgott M (2005) Multiple criteria decision analysis: state of the art surveys. Springer, New York, NY

Finke G (1977) A unified approach to reshipment, overshipment and post-optimization problems. In: Stoer J (ed) Optimization techniques. Proceedings of the 8th IFIP Conference, Part 2. Lecture Notes in Control and Transportation Sciences 7, Springer, Berlin Heidelberg New York, pp 201-208

Finke G (1983) Minimizing overshipments in bottleneck transportation problems. INFOR 21: 121-135

Fliege J, Vicente LN (2006) Multicriteria approach to bilevel optimization. Journal of Optimization Theory and Applications 131: 209-225

Ford Jr LR, Fulkerson DR (1954) Maximal flow through a network. Rand Research Memorandum 1400, The RAND Corporation, Santa Monica Subsequently published under the same title in the Canadian Journal of Mathematics 8, 1956, 399-404

Fourier JBJ (1826) Solution d'une question particulière du calcul des inégalités. Nouveau Bulletin des Sciences par la Société philomathique de Paris: 99-100

Frank A (2005) On Kuhn's Hungarian method—a tribute from Hungary. Naval Research Logistics 52: 2-5

Frisch KR (1956) La résolution des problèmes de programme linéaire par la méthode du potential logarithmique. Cahiers du Seminaire d'Économetrie 4: 7-20

Gal T (1979) Postoptimality analyses, parametric programming and related topics. McGraw Hill, New York

Gal T (1984) Linear parametric programming - A brief survey. Mathematical Programming Study **21:** 43-68

Gale D (1960) The theory of linear economic models. Mc-Graw-Hill, New York, NY

Gale D, Kuhn HW, Tucker AW (1951) Linear programming and the theory of games: Koopmans, T.C. (ed), Activity analysis of production and allocation. Cowles Commission Monograph 13: 317-329

Garey MR, Johnson DS (1979) Computers and intractability: A guide to the theory of NP-completeness. Freeman, San Francisco, CA

Garfinkel RS, Rao M (1976) Bottleneck linear programming. Mathematical Programming 11: 291-298

Gass SI, Assad AA (2005) An annotated timeline of operations research: An informal history. Kluwer, New York, NY

Gass SI, Saaty T (1955) The computational algorithm for the parametric objective function. Naval Research Logistics Quarterly 2: 39

Gauss CF (1826) Theoria combinationis observationum erroribus minimis obnoxiae. Werke, vol. 4, Göttingen, Germany

Garner Garille S, Gass SI (1981) Stigler's diet problem revisited. Operations Research 49: 1-13

Geoffrion AM (1967) Solving bicriterion mathematical programs. Operations Research 15: 39-54

Geoffrion AM, Dyer JS, Feinberg A (1972) An interactive approach for multi-criterion optimization with an application to the operation of an academic department. Management Science 19: 357-368

Gilmore PC, Gomory RE (1961) A linear programming approach to the cutting stock problem. Operations Research 9: 849-859

Gilmore PC, Gomory RE (1963) A linear programming approach to the cutting stock problem – Part II. Operations Research 11: 863-888

Gordan P (1873) Über die Auflösung linearer Gleichungen mit reellen Coefficienten. Mathematische Annalen 6: 23-28

Grattan-Guinness I (1970) Joseph Fourier's anticipation of linear programming. Operational Research Quarterly 21: 361-364

Gutin G, Punnen A (eds) (2002) The traveling salesman problem and its variations. Kluwer, Boston Dordrecht London

Hirsch WM (1957) Hirsch conjecture, verbal communication to Dantzig

Hitchcock FL (1941) The distribution of a product from several sources to numerous localities. Journal of Mathematical Physics 20**:** 224-230

Hobson RF, Weinkam JJ (1979) Curve fitting. In: Holzman AG (ed) Operations research methodology. Dekker, New York Basel, pp 335-362

Hoffman AJ (1953) Cycling in the simplex algorithm. National Bureau of Standards, Report No. 2974, Washington, D.C

Hoffman AJ, Kruskal JG (1956) Integral boundary points of convex polyhedra. In: Kuhn HW, Tucker AW (eds) Linear inequalities and related systems. Annals of Mathematics Studies, Princeton University Press, Princeton, NJ, pp 223-246

Huard P (1967) Resolution of mathematical programming with nonlinear constraints by the method of centers. In: Abadie J (ed) Nonlinear programming. North-Holland, Amsterdam, pp 209-219

Hussein ML, Al-Ghaffar FSA (1996) An interactive approach for vector optimization problems. European Journal of Operational Research 89: 185-192

Ignizio JP (1982) Linear programming in single- & multiple-objective systems. Prentice-Hall, Englewood Cliffs, NJ

Ignizio JP, Cavalier TM (1994) Linear programming. Prentice-Hall, Englewood Cliffs, NJ

Ijiri Y (1965) Management goals and accounting for control. Rand McNally, Chicago, IL

Inuiguchi M, Ramík J (2000) Possibilistic linear programming: A brief review of fuzzy mathematical programming and a comparison with stochastic programming in portfolio selection problem. Fuzzy Sets and Systems 111: 3-28

Jeroslow, RG (1973) There cannot be any algorithm for integer programming with quadratic constraints. Operations Research 21: 221-224

Jones DF, Tamiz M (2002) Goal programming in the period 1990-2000. In: Ehrgott M, Gandibleux X (eds) Multiple criteria optimization: State of the art annotated bibliographic surveys. Springer, New York, NY

Kantorovich LV (1939) (1960) Mathematical methods in the organization and planning of production. Translation of the 1939 original (in Russian). In: Management Science **6:** 366

Karmarkar N (1984) A new polynomial-time algorithm for linear programming. Combinatorica 4: 373-395

Karp RM (1972) Reducibility among combinatorial problems. In: Miller R E, Thatcher J W (eds) Complexity of computer computations. Plenum Press, New York, pp. 85-103

Khachian LG (1979) Polynomial algorithm for linear programming. Doklady Akademii Nauk SSSR 244: 1093-1096 (in Russian), translated into English in the same year in Soviet Mathematics Doklady 20: 191-194

Klee V, Minty GJ (1972) How good is the simplex algorithm. In: Shisha O (ed) Inequalities III. Academic Press, New York, pp 159-175

Klee VL, Walkup DW (1967) The *d*-step conjecture for polyhedra of dimension $d <$ 6. Acta Mathematica 117: 53-78

König D (1931) Graphok és matrixok (Graphs and matrices, in Hungarian). Mathematikai és Fizikai Lapok 38: 116-119

Koopmans TC (1951) Activity analysis of production and allocation. Wiley, New York, NY

Koopmans TC (1951a) Analysis of production as an efficient combination of activities. In: Koopmans TC (ed) Activity analysis of production and allocation. Wiley, New York, NY, pp. 33-97

Koopmans TC, Beckmann M (1957) Assignment problems and the location of economic activities. Econometrica 25: 53-76

Kuhn HW (1955) The Hungarian method for the assignment problem. Naval Research Logistics Quarterly 2: 83-97

Kuhn HW (1991) On the origin of the Hungarian method. In: Lenstra JK, Rinnooy Kan AHG, Schrijver A (eds) History of mathematical programming: a collection of personal reminiscences. CWI, Amsterdam and North Holland, Amsterdam, The Netherlands, pp 77-81

Kuhn HW, Tucker AW (1951) Nonlinear programming. In: Neyman J (ed) Proceedings of second Berkeley symposium on mathematical statistics and probability. University of California Press, Berkeley, CA, pp 481 – 491

Kuhn HW, Tucker AW (eds) (1956) Linear inequalities and related systems. Annals of Mathematics Studies 38. Princeton, NJ

Ladner RE (1977) The computational complexity of provability in systems of modal propositional logics. SIAM Journal on Computing 6: 467-480

Land AH, Doig AG (1960) An automatic method of solving discrete linear programming problems. Econometrica 28: 496-520

Lawler EL, Lenstra JK, Rinnooy Kan AHG, Shmoys DB (1985) The traveling salesman problem: A guided tour of combinatorial optimization. Wiley, Chichester, United Kingdom

Lee SM (1972) Goal programming for decision analysis. Auerbach, Philadelphia, PA

Lemke CE (1954) The dual method of solving linear programming problems. Naval Research Logistics Quarterly 1: 36-47

Lenstra JK, Rinnooy Kan AHG, Schrijver A (eds) History of mathematical programming: A collection of personal reminiscences. Elsevier, Amsterdam

Leontief WW (1936) Quantitative input and output relations in the economic system of the United States. Review of Economic Statistics 18: 105-125

Levin AYu (1965) On an algorithm for the minimization of convex functions. Soviet Mathematics Doklady 6: 286-290

Lootsma FA (1972) A survey of methods for solving constrained minimization problems via unconstrained minimization. In: Lootsma FA (ed) Numerical methods for non-linear optimization. Academic Press, London

Mangasarian OL (1969) Nonlinear programming. McGraw-Hill Publ. Co., New York, NY. Reprinted as SIAM Classic in applied mathematics 10, 1994, Philadelphia, PA

Manne AS (1953) Notes on parametric linear programming. Rand Paper P- 468. The Rand Corporation, Santa Monica, CA

Marglin JA (1967) Public investment criteria. MIT, Cambridge, MA

Markowitz HM (1952) Portfolio selection. Journal of Finance 7: 77-91

McCloskey JF (1987) The beginnings of operations research: 1934-1941. Operations Research 35: 143-152

McMullen P (1970) The maximum numbers of faces of a convex polytope. Mathematika 17: 179-184

Moore JH, Weatherford LR (2001) Decision modeling with Microsoft® Excel. Prentice Hall, Englewood Cliffs, NJ

Motzkin TS (1936) Beiträge zur Theorie der linearen Ungleichungen. Doctoral Thesis, Zürich, Switzerland

Murty KG (1986) The gravitational method for linear programming. Opsearch 23: 206-214

Murty KG (1992) Network programming. Prentice Hall, Englewood Cliffs, NJ

Nemhauser GL, Wolsey LA (1988) Integer and combinatorial optimization. Wiley, New York, NY

Neumann J, von Morgenstern O (1944) Theory of games and economic behavior. Princeton University Press, Princeton, NJ

Nicholson K (2006) Linear algebra with applications. 5th edn. McGraw Hill, Boston, MA

Norback JP, Morris JG (1980) Fitting hyperplanes by minimizing orthogonal deviations. Mathematical Programming 19: 102-105

Orchard-Hays W (1955) Notes on linear programming (Part 6) The RAND vode for the simplex method (SX4). Report #1440, The RAND Corporation, Santa Monica, CA. (Based on the author's unpublished Master's Thesis in 1952)

Orden A (1952) Application of the simplex method to a variety of matrix problems. In: Orden A, Goldstein L (eds) Proceedings of the Symposium on linear inequalities and programming. pp 28-50

Overbay, S., Schorer, J., Conger, H. http://www.ms.uky.edu/~carl/ma330/project2/al-khwa21.html, accessed 4/25/2007

Padberg M (1995) Linear optimization and extensions. Springer, Berlin Heidelberg New York

Pareto V (1906) Manuale di econnomia politica. Società Editrice Libraria, Milan, Italy

Parisot GR (1961) Résolution numérique approchée du problème de programmation linéaire par application de la programmation logarithmique. Revue française de la recherché opérationnelle 20: 227-259

Parkinson CN (1957) Parkinson's law. Houghton Mifflin, Boston, MA

Pawlak Z (1991) Rough sets: Theoretical aspects of reasoning about data. Kluwer, Dordrecht

Philip J (1972) Algorithms for the vector maximization problem. Mathematical Programming 2: 207-229

Rardin RL (1998) Optimization in operations research. Prentice-Hall, Upper Saddle River, NJ

Reeves GR, Franz LS (1985) A simplified interactive multiple objective linear programming procedure. Computers & Operations Research 12: 589-601

Reeves GR, Hedin SR (1993) A generalized interactive goal programming procedure. Computers & Operations Research 20: 247-253

Renegar J (1988) A polynomial-time algorithm based on Newton's method for linear programming. Mathematical Programming 40: 59-93

Roos C, Terlaky T, Vial J-Ph (2006) Interior point methods for linear optimization. 2nd edn. Springer, New York, NY

Saigal R (1995) Linear programming. Kluwer, Boston, MA

Sakawa M (2002) Fuzzy multiobjective and multilevel optimization. In Ehrgott M., Gandibleux X. (eds) Multiple criteria optimization: state of the art annotated bibliographic survey, Kluwer, The Netherlands, pp 171-226

Schmalenbach E (1948) Pretiale Wirtschaftslenkung. Band 1: Die optimale Geltungszahl, Dresden, Germany

Schniederjans MJ (1984) Linear goal programming. Petrocelli Books, Princeton, NJ

Schniederjans M (1995) Goal programming: methodology and applications. Kluwer, Boston, MA

Shin WS, Ravindran A (1991) Interactive multiobjective optimization survey I: continuous case. Computers & Operations Research 18: 97-114

Shor NZ (1970) Utilization of the operation of space dilation in the minimization of convex functions. Kibernetika 1: 6-12 (in Russian), translated into English in the same year in Cybernetics 6: 7-15

Sierksma G (2002) Linear and integer programming: theory and practice (2nd ed.) Marcel Dekker, New York, NY

Simmons DM (1972) Linear programming for operations research, Holden-Day, San Francisco, CA

Simonnard M (1966) Linear programming. Prentice-Hall, Englewood Cliffs, NJ

Simon HA (1957) Models of man. Wiley, Yew York, NY

Stadler W (1984) A comprehensive bibliography on multicriteria decision making. In: Zeleny M (ed) MCDM past decade and future trends. JAI Press, CT, pp 223-328

Steuer RE (1986) Multiple criteria optimization: theory, computation, and application. Wiley, New York, NY

Steuer R E (1995) The ADBASE multiple objective linear programming package. In Gu J, Chen J, Wei Q, Wang S (eds), Multiple criteria decision making, SCI-TECH, Windsor, England, pp 1-6

Stigler G (1945) The cost of subsistence. Journal of Farm Economics 25: 303-314

Strassen V (1969) Gaussian elimination is not optimal. Numerische Mathematik 13: 354-356

Sun M, Stam A, Steuer RE (1996) Solving multiple objective programming problems using feed-forward artificial neural networks: The interactive FFANN procedure. Management Science 42: 835-849.

Swarc W (1971) The transportation paradox. Naval Research Logistics Quarterly 18: 185-202

Tanino T, Tanaka T, Inuiguchi M (eds) (2003) Multi-objective programming and goal programming: theory and applications. Advances in soft computing 21, Springer, Berlin Heidelberg

Trzaskalik T, Michnik J (eds) (2002) Multiple objective and goal programming. Advances in soft computing 12, Springer, Berlin Heidelberg

Vanderbei RJ (2001) Linear programming: foundations and extensions. 2nd edn. Kluwer, Boston, MA

Vasant P, Nagarajan R, Yaacob S (2005) Fuzzy linear programming with vague objective coefficients in an uncertain environment. Journal of the Operational Research Society 56: 597-603

Vicente LN (2001) Bilevel programming: introduction, history, and overview. In: Floudas CA, Pardalos PM (eds) Encyclopedia of optimization. Springer, New York

Vicente LN, Calamai PH (1994) Bilevel and multilevel programming: A bibliographic review. Journal of Global Optimization 5: 291-306

von Neumann J (1947) On a maximization problem (manuscript). Institute for Advanced Study, Princeton, NJ, Nov 1947

von Stackelberg H (1934) Marktform und Gleichgewicht. Springer, Vienna & Berlin

Walford, R.L. (2007) http://yarrow.best.vwh.net/Usda_data/foods_db.html accessed April 24, 2007

Yu PL (1973) Introduction to domination structures in multicriteria problems. In: Cochrane JL, Zeleny M (eds.) Multiple criteria decision making, University of South Carolina Press, Columbia SC, pp 249-261

Zadeh M (1963) Optimality and non-scalar-valued performance criteria. IEEE Transactions on Automatic Control AC 8: 59-60

Zadeh LA (1979) Fuzzy sets. In: Holzman AG (ed) Operations research methodology. Marcel Dekker, New York Basel, pp 569-606

Zeleny M (1973) Compromise programming. In: Cochrane JL, Zeleny M (eds), Multiple criteria decision making. University of South Carolina Press, Columbia, SC, 262-301

Zhang S (1991) On anti-cycling rules for the simplex method. Operations Research Letters 10: 189-192

Zimmermann H-J (1976) Description and optimization of fuzzy systems. International Journal of General Systems 2: 209-215

Zimmermann H-J (1978) Fuzzy programming and linear programming with several objectives. Fuzzy Sets and Systems 1: 45-55

Zionts S (1969) The criss-cross method for solving linear programming problems. Management Science 15: 426-445

Zykina AV (2004) A generalized solution of an interactive goal programming problem. Cybernetics and Systems Analysis 40: 277-283

SUBJECT INDEX

A

absolute values (of constraints) 299-301
Absolute values (objective function) 301-306
affine linear combination 21-22
affine scaling method 274, 277-284
allocation problems 71-75
alternative optimal solutions 140
antisymmetric matrix 170
artificial objective function 153-154
artificial variables 63
aspiration levels 351
assignment problems 102-107

B

balanced transportation problem 93
barrier method 285-293
basic feasible point 28
basic feasible solution 144
basic point 28
basic solution 144
basic variable 9
basis matrix 9
basis of a matrix 9
bilevel programming 359-362
binary encoding 35
binding relation 20
bi-objective programming 350
bisection search method 39
blending problems 89-91
block-angular structure 95
bottleneck objective 306-313
bounce method 267
bounded set 20
break-even analysis 58-60

C

canonical form 143
case study 111-128
characteristic value 3
Chebyshev metric 343
circling 164
closed-form solution 31
closed set 18
cold start 225
column vector 1
commensurability 353
compact set 20
column generation 219-224
constraint method 339-341
constraints, loosen up 139
constraints, too tight 138
convex function 301
convex hull 29
convex polyhedral cone 29
convex set 23-30
Cramer's rule 10
curve fitting 304-305
cutting stock problems 75-80, 221-224
cycling 165

D

data envelopment analysis (DEA) 82-85, 321-323
decision problems 41
degeneracy in postoptimality analyses 245-248
destinations 92
deterministic model 52
determinant of a matrix 2
deviational variables 302-305, 352-359
diet problems 67-70, 199-200
direction of the objective function 131

dual degeneracy 140, 160-163
dual price 64
dual problem 178, 183-197
dual simplex method 203-212

E

efficient set 327
eigenvalue 3
ellipsoid method 268-272
employee scheduling 80-82
entering variable 145
essential relation 20
Euclidean metric 343
excess variables 63
exponential algorithm 36
external pivoting 264-267
extreme point 28

F

Farkas' lemma 168
feasible set 19
fractional programming 320-323
fuzzy membership function 347
fuzzy programming 346-351

G

Gauss-Jordan pivoting method 10-18
generalized assignment problem 106-107
generalized upper bounding technique 212
global optimum 137
goal programming 351-359
gradient of the objective function 131
gravitational method 267
guillotine cuts 80

H

halfspace 18
Hirsch conjecture 138
hundred percent rule 240, 245
Hungarian method 102, 105-106
hyperbolic programming 320-323
hyperplane 18

I

idempotent matrix 294
identity matrix 1
improvement cone 330
improving feasible direction 134
indicators 64, 145
indifference region 346
interior point method 273-294
intersection of convex sets 27
intersection of fuzzy sets 348
inventory balancing constraints 86
inventory model 85-89
inverse of a matrix 2
iso-profit lines 130

L

Lapace rule to evaluate determinants 2-3
leader-follower scenario 360
leaving variable 145
left shadow price 238
lexicographic method 339
lexicographic selection rule 165
linear combination 21
linear convex combination 22
linear dependence 7
linear relation 5
local optimum 137
lower bounding constraints 295-296

M

Manhattan metric 343
matrix 1
matrix search method 39
Maximin problem 313-320
Minimax problem 313-320
Minkowski-Farkas lemma 170
Minkowski metrics 343
more-for-less paradox 101
multi-attribute decision making 325
multicriteria decision making 325
multiobjective programming 325-362
multiobjective simplex method 336

N

Newton-Raphson method 3-4
nonbasic variable 9
nondominated frontier 334
nondominated solution (set) 327
noninferior solution (set) 327
nonnegative linear combination 21
nonnegative orthant 19
nonnegativity constraints 62
normal form (linear program) 63
NP-complete problems 42-43
NP-hard problems 42-43

O

objective space 335
opportunity cost 64, 198
optimale Geltungszahl 167
origins 92
overachievements 352
overshipments 101-102, 201-202

P

pairwise exchange method 41
parametric programming 260
Pareto optimal solution (set) 327
perturbation function 236-239, 244, 248
perturbation technique 165
phase 1 154
phase 2 148
pivoting 10-18, 150-151
pivot column 148
pivot row 148
polyhedron 20
polynomial algorithm 36
polytope 20
portfolio selection problems 73-75
possibilistic programming 351
postoptimality analysis 225-260
preemptive priorities 339
primal degeneracy 142, 163-165
primal-dual relations 179, 183-197
primal problem 178, 183-197
primal simplex method 143-157
printout 65-66, 254-255
probabilistic model 52
production-inventory model 107-128
product of two matrices 2
projection matrix 294
projective scaling method 273
proportionality 6, 53
proxy criterion 50

Q

quadratic assignment problem 107

R

rank of a matrix 8
recognition problems 41
reduced cost 64, 145
redundant relation 20
reference point programming 341-346
reshipments 99-101, 201-202
right shadow price 238

robust model 227
rough sets 351
row vector 1

S

scalar 1
search in an unordered set 40
sensitivity analysis 225-260
sequential scanning method 39
set of feasible solutions 19
shadow price 64, 198
simplex 4
simplex multipliers 64
size of the instant of a problem 35
skew symmetric matrix 170
slack variables 63
stalling 164
standard form 143
stochastic model 52
strong complementary slackness 174
strong duality theorem 172
sum of matrices 1
surplus variables 63
surrogate criterion 50
symmetric matrix 2, 294
system of simultaneous linear equations, 5-23

T

target values 351
theorems of the alternative 168
tight relation 20
time complexity function 36
trace of a matrix 2
tradeoff curves 336
transportation problems 91-102, 200-202
transpose of a matrix 2
transshipment points 99
traversal method 262-264
trim loss problems 75-80
two-phase method 155-158

U

underachievements 352
union of fuzzy sets 348
unit vector 1
unrestricted variables 296-298
upper bounding technique 212-218

V

vector optimization 327-336
volume of a simplex 4

W

warm start 248
weak complementary slackness 173
weak duality theorem 171-172
weighting method 337-339